



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیووتر

پروژه کارشناسی

گرایش معماری سیستم‌های کامپیووتری

پیاده‌سازی سیستم لرزه‌نگار جهت نگهداری و تعمیرات  
پیش‌بینانه تجهیزات بر بستر اینترنت اشیاء

نگارنده

آریان بوکانی

استاد راهنما

دکتر حمیدرضا زرندی

۱۴۰۲ تیر

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

# صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

## نکات مهم:

- نگارش پایان نامه/رساله باید به **زبان فارسی** و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت **پشت و رو(دورو)** بلامانع است و انجام آن توصیه می شود.



دانشگاه صنعتی امیرکبیر  
(پلی‌تکنیک تهران)

به نام خدا

تاریخ: تیر ۱۴۰۲

## تعهدنامه اصالت اثر

اینجانب آریان بوکانی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظرارت و راهنمایی استاد دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مأخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است. در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

آریان بوکانی

امضا

تَعْدِيمٌ

آنکه جز به فضلش امیدی نیست...

# پاسکزاری

از استاد بزرگوارم جناب آقای دکتر حمیدرضا زرندی که با حسن خلق و گشاده روبی، رهنمودهای روشنگر خود را برای انجام این پروژه از من دریغ نکرده‌اند، از استاد مشاورم جناب آقای دکتر حامد فربه که راهنمایی‌های ایشان از بد و ورود به دانشگاه کمک بسیاری به من در طی کردن مسیر تحصیل بوده‌اند، از مادر و پدرم که همواره در مواجهه با سختی‌های این دنیا دلسوزانه همراه‌هم بوده‌اند، و از سایر عزیزانی که در کنارشان این نتیجه حاصل آمد کمال تشکر و قدردانی را دارم.

آریان بوکانی  
سیر ۱۴۰۲

## چکیده

در سال‌های اخیر اینترنت اشیاء به یکی از داغ‌ترین موضوعات فناوری تبدیل شده است. کاربرد این فناوری در تمامی حوزه‌های زندگی انسان و همچنین پیشرفت‌های اخیر در حوزه‌های جمع‌آوری داده، شبکه و هوش مصنوعی باعث شده‌اند که اینترنت اشیاء مورد توجه محققان قرار گیرد. یکی از چالش‌های موجود در صنایع و کارخانه‌ها تعویض بهینه قطعات است. با توجه به نبود اطلاعات کافی برای تحلیل وضعیت قطعات، راه حل مناسب برای اطمینان از کارکرد خط تولید، استفاده از متخصصان جهت بازرگانی از وضعیت تجهیزات یا تعویض آنها بدون توجه به وضعیت فعلی و صرفاً طبق زمان‌بندی قبلی است. این راه حل‌ها علاوه بر نداشتن دقت لازم، هزینه و زمان زیادی بر صنایع تحمیل می‌کنند. در این پروژه قصد داریم که با استفاده از ترکیب اینترنت اشیاء و هوش مصنوعی، چارچوبی برای تحلیل داده‌های لرزش تجهیزات ارائه گردد تا با استفاده از آن بتوان عمر مفید باقیمانده قطعات را بخوبی پیش‌بینی کرد. همچنین با استفاده از درگاه مبتنی بر وب، نتایج حاصله به افراد متخصص نشان داده شود تا بتوانند برنامه‌ریزی دقیقی برای نگهداری و تعمیر تجهیزات ارائه کنند. با بکارگیری این رویکرد هزینه و وقت صرف شده برای تعویض قطعات به شکل چشمگیری کاهش می‌یابد.

## واژه‌های کلیدی:

اینترنت اشیاء، نگهداری پیش‌بینانه، اینترنت اشیاء صنعتی، نگهداری سیستم سایبری-فیزیکی

# فهرست مطالب

صفحه	عنوان
۱	۱ مقدمه
۲	۱-۱ تعمیر و نگهداری
۳	۲-۱ راهکار پیشنهادی
۳	۳-۱ کارهای مشابه
۵	۲ مفاهیم پایه و تجهیزات
۶	۱-۲ حسگر
۷	۲-۲ گره انتهایی
۸	۳-۲ پروتکل ارتباطی
۸	۱-۳-۲ استاندارد زیگبی
۹	۲-۳-۲ ویژگی‌های زیگبی
۱۱	۳-۳-۲ دسته‌بندی دستگاهها
۱۲	۴-۳-۲ توپولوژی
۱۳	۵-۳-۲ دیجی بین‌الملل
۱۳	۶-۳-۲ طرز کار دستگاه‌های ایکس‌بی
۱۵	۷-۳-۲ ارتباط سریال ماژول‌های ایکس‌بی
۱۷	۴-۲ دروازه
۱۸	۵-۲ پایگاه داده‌ی سری زمانی
۱۹	۶-۲ پیش‌پردازش
۲۰	۷-۲ برنامه‌ی تحت وب
۲۱	۸-۲ جمع‌بندی و نتیجه‌گیری
۲۲	۳ پیاده‌سازی سیستم
۲۳	۱-۳ معماری کلی سیستم
۲۳	۲-۳ گره انتهایی و آردوبینو
۲۴	۱-۲-۳ خواندن از حسگر و نمونه‌برداری

۲۵	.....	۲-۲-۳ تشكيل فريم زيگبي و نوشتن در درگاه سريال
۲۸	.....	۳-۲-۳ خوابيدن و بيدارشدن در بازه‌های زمانی مشخص
۲۹	.....	۳-۳ دروازه
۲۹	.....	۱-۳-۳ فايل env_vars.py
۲۹	.....	۲-۳-۳ فايل models.py
۳۰	.....	۳-۳-۳ فايل receiver.py
۳۰	.....	۴-۳-۳ فايل requirements.txt
۳۱	.....	۵-۳-۳ فايل schedule.sh
۳۱	.....	۶-۳-۳ فايل send_data.py
۳۱	.....	۷-۳-۳ فايل signup.py
۳۱	.....	۸-۳-۳ فايل utils.py
۳۱	.....	۴-۳ پيشپردازش
۳۲	.....	۵-۳ پايگاه داده سري زمانی
۳۳	.....	۱-۵-۳ ساختار کلي
۳۴	.....	۲-۵-۳ اندازه‌گيری‌ها و توابع مربوط به نوشتن و گرفتن داده
۳۴	.....	۳-۵-۳ توابع گرفتن شناسه گره و اندازه‌گيری
۳۴	.....	۴-۵-۳ تابع get_starting_service_date
۳۵	.....	۵-۵-۳ تابع حذف داده‌ها و حافظه پنهان
۳۵	.....	۶-۳ برنامه وب و چارچوب ويوب
۳۵	.....	۱-۶-۳ کتابخانه‌ها و ماژول‌ها
۳۷	.....	۲-۶-۳ ساختار پروژه
۴۰	.....	۳-۶-۳ مؤلفه‌های اصلی
۴۵	.....	۷-۳ نتیجه‌گيری و جمع‌بندی
۴۶	.....	۴ چالش‌ها و محدودیت‌ها
۴۷	.....	۱-۴ چالش‌های سخت‌افزاری
۴۷	.....	۲-۴ چالش‌های نرم‌افزاری
۴۸	.....	۳-۴ جمع‌بندی و نتیجه‌گيری

۴۹	.....	۵	جمع‌بندی و نتیجه‌گیری و پیشنهادها
۵۰	.....	۱-۵	۱-۵ جمع‌بندی و نتیجه‌گیری
۵۰	.....	۲-۵	۲-۵ پیشنهادها
۵۰	.....	۱-۲-۵	۱-۲-۵ حل محدودیت‌های سخت‌افزاری
۵۱	.....	۲-۲-۵	۲-۲-۵ پیاده‌سازی ارسال اعلان و برنامه تلفن همراه
۵۱	.....	۳-۲-۵	۳-۲-۵ تکمیل کتابخانه ایکس‌بی
۵۱	.....	۴-۲-۵	۴-۲-۵ استفاده از سایر معیارهای محیطی
۵۲	.....	منابع و مراجع	
۵۵	.....	پیوست	

# فهرست اشکال

شکل

صفحه

۱-۱ سیاست‌های انجام تعمیرات و نگهداری [۱]	۲
۱-۲ حسگر مدل ADXL 345	۷
۲-۱ بورد آردوینو نانو	۷
۳-۱ مقایسه پروتکل زیگبی و سایر پروتکل‌های اینترنت اشیاء	۹
۴-۱ شبکه زیگبی شامل هماهنگ‌کننده، مسیریاب و گره‌پایانی [۱۰]	۱۲
۵-۱ توپولوژی‌های مختلف شبکه زیگبی [۱۱]	۱۳
۶-۱ ماژول زیگبی مدل XBee S2	۱۴
۷-۱ فرایندهای لازم برای ارتباط دو ماژول ایکس‌بی [۱۲]	۱۴
۸-۱ ارتباط سریال در حالت شفاف [۱۲]	۱۵
۹-۱ ارتباط سریال در حالت API [۱۲]	۱۷
۱۰-۱ بورد زبری‌پایی ۴ مدل بی	۱۸
۱۱-۱ پلتفرم متن باز پایگاه داده سری زمانی InfluxDB [۱۵]	۱۹
۱۲-۱ نمای کلی سیستم	۲۴
۱۳-۱ ساختار فایل کتابخانه ایکس‌بی	۲۶
۱۴-۱ ساختار کلی فریم API [۱۲]	۲۶
۱۵-۱ ساختار پروژه دروازه	۲۹
۱۶-۱ ساختار کلی پایگاه داده سری زمانی	۳۳
۱۷-۱ کد پرس‌وجوی vibration_measurement	۳۴
۱۸-۱ ساختار پروژه برنامه وب	۳۸
۱۹-۱ مؤلفه کارت خلاصه	۴۰
۲۰-۱ صفحه ورود و پیغام خطای رمز عبور نادرست	۴۱
۲۱-۱ صفحه ثبت‌نام کاربر جدید	۴۱
۲۲-۱ صفحه دروازه‌های فعال	۴۲
۲۳-۱ صفحه داشبورد	۴۳
۲۴-۱ نمودار ریشه میانگین مریع	۴۳

## فهرست اشکال

- ۱۴-۳ نمودار چگالی طیفی توان ..... ۴۴
- ۱۵-۳ نمودار سری زمانی ارتعاش ..... ۴۴

## فهرست جداول

صفحه

جدول

- ۱-۲ مقایسه بین دو نوع حسگر مبتنی بر پیزوالکتریک و MEMS [۵] ..... ۶
- ۱-۳ بخش‌های مختلف داده‌های فریم درخواست ارسال [۱۲] ..... ۲۸

## فهرست نمادها

نماد		مفهوم
$K$		تعداد همهی نمونه‌های موجود در یک اندازه‌گیری
$n$		گره $n$ ام
$m$		اندازه‌گیری $m$ ام
$k$		نمونه‌ی $k$ ام در یک اندازه‌گیری
$a_{nmk}$		بردار سهبعدی مربوط به اندازی‌گیری لرزش
$\hat{a}$		بردار هنجارشده $a$
$g$		شتاب گرانش زمین

مقدمه

فصل اول

## ۱-۱ تعمیر و نگهداری

در دنیای نوین امروز، وابستگی به فناوری و صنایع مختلف برای ارائه خدمات و تولید محصولات مورد نیاز بشدت افزایش یافته است. در نتیجه این افزایش، تعداد کارخانه‌ها و تجهیزات برای رفع این نیازها نیز افزایش یافته است. همه تجهیزات و وسایل پس از استفاده زیاد دچار نقص و فراوانی می‌شوند. خرابی تجهیزات در کارخانه‌ها و خطوط تولید پدیده‌ای نامطلوب است. خرابی تجهیزات باعث ایست خطر تولید و وارد کردن ضرر مالی به صاحبان کارخانه‌ها خواهد شد. بهمین دلیل، تعمیر و نگهداری تجهیزات اهمیت بالایی دارد.

سیاست‌های انجام تعمیرات و نگهداری به سه دسته اصلی تقسیم می‌شوند؛ که در [شکل ۱-۱](#) آورده شده است:



شکل ۱-۱: سیاست‌های انجام تعمیرات و نگهداری [\[۱\]](#)

این سیاست‌ها عبارتند از:

- **نگهداری اصلاحی<sup>۱</sup>**: این نوع نگهداری با اتفاق افتادن خرابی رخ می‌دهد و تنها در صورت خرابی قطعه تعمیر انجام می‌شود. همانطور که مشخص است در این حالت روند کارکرد عادی متوقف می‌شود: به علاوه، در این حالت تعمیر می‌تواند با تأخیر انجام شود زیرا ممکن است در لحظه قطعات مورد نیاز برای تعمیر موجود نباشد. از این سیاست بیشتر به عنوان مکملی برای سیاست‌های بعدی استفاده می‌شود [\[۱\]](#).

- **نگهداری پیش‌گیرانه<sup>۲</sup>**: در این حالت، نگهداری و تعمیر بر اساس رابطه بین نرخ خرابی، توزیع زمان خرابی و سایر آستانه‌هایی که از تعداد زیادی از آمارهای خرابی بدست می‌آید زمان‌بندی می‌شود. در واقع در این نوع نگهداری هدف کاهش احتمال خرابی قطعات است. این نوع نگهداری

<sup>1</sup>Corrective Maintenance

<sup>2</sup>Preventive Maintenance

با توجه به نوع اطلاعات به سیاست‌های وابسته عمر،<sup>۳</sup> سیاست‌های دوره‌ای<sup>۴</sup>، سیاست‌های ترتیبی<sup>۵</sup> و سیاست‌های محدود‌کننده خرابی<sup>۶</sup> تقسیم می‌شوند<sup>[۱]</sup>.

- نگهداری پیش‌بینانه<sup>۷</sup>: این نوع نگهداری بر روند کاهش کارایی قطعات با استفاده از وضعیت آنها نظارت کرده، وضعیت آنها را در آینده پیش‌بینی می‌کند و مداوم برنامه تعمیر و نگهداری را بروزرسانی می‌کند، تا اینکه شرایط توقف بروزرسانی برآورده شود. با توجه به این ویژگی، این سیاست فقط برای تجهیزاتی که وضعیت آنها قابل نظارت است می‌تواند استفاده شود<sup>[۱]</sup>.

## ۲-۱ راهکار پیشنهادی

با توجه به اهمیت نگهداری و نقش نگهداری اصولی در کاهش هزینه، در این پژوهه به ایجاد چارچوبی مناسب برای ارائه راهکاری بر مبنای سیاست نگهداری پیش‌بینانه می‌پردازیم. یکی از نیازهای این سیاست حجم زیادی از داده جمع‌آوری شده از وضعیت تجهیزات برای پیش‌بینی دقیق است. در این پژوهه در بستر اینترنت اشیاء<sup>۸</sup> داده‌های لرزش با استفاده از سنسور MEMS<sup>۹</sup> جمع‌آوری شده و در زمان‌های مشخص با استفاده از پروتکل زیگبی<sup>۱۰</sup> برای دروازه<sup>۱۱</sup> فرستاده می‌شود. پس از جمع‌آوری حجم مشخصی از داده‌ها اطلاعات برای پردازش بیشتر به سرور فرستاده می‌شوند. در انتها نیز اطلاعات پردازش شده و تخمین عمر مفید باقیمانده<sup>۱۲</sup> در یک صفحه وب نمایش داده می‌شود.

## ۳-۱ کارهای مشابه

در [۲] مدلی برای خرابی قطعات مبتنی بر میزان استفاده از تجهیزات و بار داخلی آنها ارائه شده است اما مدل ارائه شده محدود به یک مدل تجهیزات است و برای استفاده در سایر مدل‌ها نظارت مجدد لازم است. در [۳] رویکردی مبتنی بر شبکه عصبی جهت پیش‌بینی عمر مفید باقیمانده تجهیزات چرخی

<sup>3</sup>Age Dependent Strategies

<sup>4</sup>Periodic Strategies

<sup>5</sup>Sequential Strategies

<sup>6</sup>Failure Limited Strategies

<sup>7</sup>Predictive Maintenance

<sup>8</sup>Internet of Things

<sup>9</sup>Micro Electric Mechanical Sensor

<sup>10</sup>Zigbee Protocol

<sup>11</sup>Gateway

<sup>12</sup>Remaining Useful Lifetime(RUL)

ارائه شده است اما تنها مختص به این دسته از تجهیزات است. در [۴] رویکردی مبتنی بر شبکه عصبی جهت پیش‌بینی زمان نگهداری و تعمیرات تجهیزات بر اساس مدل خرایی و کارایی آنها ارائه می‌شود اما در یک محیط شبیه‌سازی شده و کنترل شده آزمایش شده است و در هنگام استفاده در محیط واقعی غیرعملی است.

بطور کلی در پژوهش‌های یادشده رویکردهای ارائه شده مختص نوع خاصی از تجهیزات است و در یک محیط آزمایشگاهی و کنترل شده ارزیابی شده‌اند. در حالیکه در این پژوهش با استفاده از داده‌های جمع‌آوری شده از قطعات مختلف سعی کرده‌ایم رویکردی کلی و مناسب محیط واقعی و صنعتی ارائه دهیم.

## فصل دوم

# مفاهیم پایه و تجهیزات

در این فصل به بررسی مفاهیم و تجهیزاتی که در این پژوهه استفاده شده است می‌پردازیم. در هر بخش دلیل انتخاب خود را شرح می‌دهیم و آن را با سایر راه حل‌ها مقایسه می‌کنیم.

## ۱-۲ حسگر

همانطور که گفته شد برای پیش‌بینی نگهداری و عمر مفید تجهیزات نیاز به نظارت و جمع‌آوری اطلاعاتی راجع به این تجهیزات داریم. در این پژوهه، اطلاعات جمع‌آوری شده مربوط به وضعیت لرزش تجهیزات است که با استفاده از حسگرهای لرزش MEMS اندازه‌گیری می‌شوند.

در این پژوهه تصمیم گرفتیم که بجای دما یا رطوبت از لرزش تجهیزات برای پیش‌بینی استفاده کنیم؛ زیرا لرزش برخلاف دو معیار دیگر بطور مستقیم شرایط عملیاتی تجهیزات را منعکس می‌کند که بیشتر بر اساس رفتارهای مکانیکی مانند چرخش موتور یا حرکت جریان در لوله ایجاد می‌شوند. هم‌چنین دو معیار دیگر وابستگی زیادی به محیط دارند اما لرزش تقریباً مستقل از عوامل خارجی است [۵]. برای اندازه‌گیری لرزش دو نوع حسگر لرزش وجود دارد. حسگرهای مبتنی بر شتاب‌سنج MEMS بر اکثر محدودیت‌های حسگرهای قدیمی مبتنی بر پیزوالکتریک<sup>۱</sup> غلبه می‌کنند. تفاوت این دو حسگر را می‌توان در جدول ۱-۲ [۵] دید.

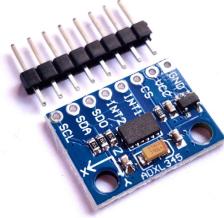
جدول ۱-۲: مقایسه بین دو نوع حسگر مبتنی بر پیزوالکتریک و MEMS

MEMS	پیزوالکتریک	
۱۰+	۳۰۰+	قیمت (دلار)
۳	۲۷	توان مصرفی (mW)
$0.05 \times 0/2 \times 0/2$	$1 \times 0.98 \times 1.97$	اندازه (اینچ)
۴۰۰۰	۷۰۰	تراکم نویز ( $\mu g$ )
۱۰۰	۱۰	بازه شتاب (g)

بطور کلی حسگرهای نوع MEMS ارزان‌تر، کم‌صرف‌تر و کوچک‌تر هستند. در این پژوهه حسگرهای با باتری کار خواهند کرد و برای اندازه‌گیری به قطعات مورد نظر متصل می‌شوند. بنابراین کم‌صرف‌بودن، ارزانی و کوچک‌بودن برای اتصال به بدنه ویژگی‌های مطلوبی است که در حسگرهای نوع MEMS یافت می‌شود. شکل ۱-۲ حسگر مدل ADXL 345 که در این پژوهه استفاده کردہ‌ایم.

این حسگر می‌تواند شتاب را در سه جهت، در چهار بازه قابل تنظیم ۲، ۴، ۸ و ۱۶ برابر گرانش با

<sup>۱</sup>Piezoelectric

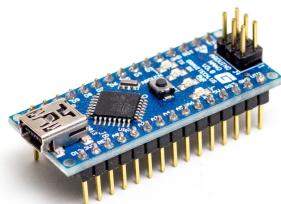


شکل ۲-۱: حسگر مدل ADXL 345

دقتهای متفاوت اندازه‌گیری کند. خروجی این حسگر نیز با دو پروتکل SPI<sup>۲</sup> و I2C قابل انتقال است.

## ۲-۲ گره انتهایی

برای کاهش بیشتر مصرف انرژی، دریافت اطلاعات حسگر و فرستادن آن برای دروازه به یک بورد<sup>۳</sup> نیاز داریم. بوردهای آردوبینو<sup>۴</sup> اگرچه محدودیتهایی دارند، اما با توجه به ارزان بودن و برآورده کردن نیاز ما انتخاب مناسبی هستند. بورد استفاده شده در این پروژه آردوبینو نانو است که در [شکل ۲-۲](#) قابل مشاهده است. در این بورد از پورت سریال<sup>۵</sup> برای ارتباط با ماژول زیگبی<sup>۶</sup> و از پروتکل I2C برای ارتباط با حسگر استفاده می‌کنیم. یکی از محدودیتهای این بورد حافظه کم است که در نرخ‌های بالای نمونه برداری حسگر کار ما را سخت می‌کند که در فصل بعد بیشتر به آن می‌پردازیم.



شکل ۲-۲: بورد آردوبینو نانو

<sup>2</sup>Serial Peripheral Interface

<sup>3</sup>Board

<sup>4</sup>Arduino

<sup>5</sup>Serial Port

<sup>6</sup>Zigbee Module

## ۳-۲ پروتکل ارتباطی

برای ارسال اطلاعات از گره‌های متصل به حسگر به دروازه نیاز به یک پروتکل ارتباطی داریم. نیازهای ما در رابطه با این پروتکل عبارتند از:

- کم‌صرف‌بودن: با توجه به اینکه گره‌های ما به باتری متصل هستند، به پروتکلی نیاز داریم که در مصرف انرژی بسیار بهینه عمل کند تا بتوان بدون تعویض باتری اطلاعات را تا چند سال برای دروازه فرستاد.
- پوشش و نفوذ سیگنال: در هر کارخانه یک دروازه نصب شده است، کارخانه‌ها مساحتی متوسط دارند و مانع خاصی نیز در آنها وجود ندارد. پس نیاز به پوشش یا قدرت نفوذ بالایی در سیگنال ارسالی نداریم.
- توپولوژی<sup>۵</sup>: با توجه به وجود یک دروازه و چندین گره که همگی اطلاعات را برای دروازه ارسال می‌کنند، نیاز به پروتکلی داریم که از توپولوژی ستاره پشتیبانی کند.
- ارتباط مطمئن<sup>۶</sup>: ارسال ناقص و یا از دستدادن اطلاعات حسگرها می‌تواند باعث پیش‌بینی اشتباه شود. بنابراین، به پروتکلی با قابلیت ارتباط مطمئن و کنترل خطای نیاز داریم.

در شکل ۳-۲ پروتکل‌های ارتباطی با معیارهای مختلف مقایسه شده‌اند. بر اساس ویژگی‌های موردنیاز پروتکل زیگبی انتخاب مناسب‌تری نسبت به سایر پروتکل‌ها است که در ادامه آن را معرفی می‌کنیم.

## ۱-۳-۲ استاندارد زیگبی

زیگبی یکی از استانداردهای فرستنده-گیرنده<sup>۹</sup> در شبکه‌های حسگر بی‌سیم<sup>۱۰</sup> است که بصورت گستردگی استفاده قرار گرفته است. زیگبی بر روی IEEE802.15.4 مشخصات شبکه شخصی بی‌سیم با نرخ داده پایین<sup>۱۱</sup> را تعریف می‌کند تا بتواند از دستگاه‌های نظارتی و کنترلی با توان مصرفی و نرخ داده پایین پشتیبانی کند. زیگبی توسعه اتحاد زیگبی<sup>۱۲</sup> داده شده است؛ که صدها عضو دارد. اتحاد زیگبی

<sup>7</sup>Topology

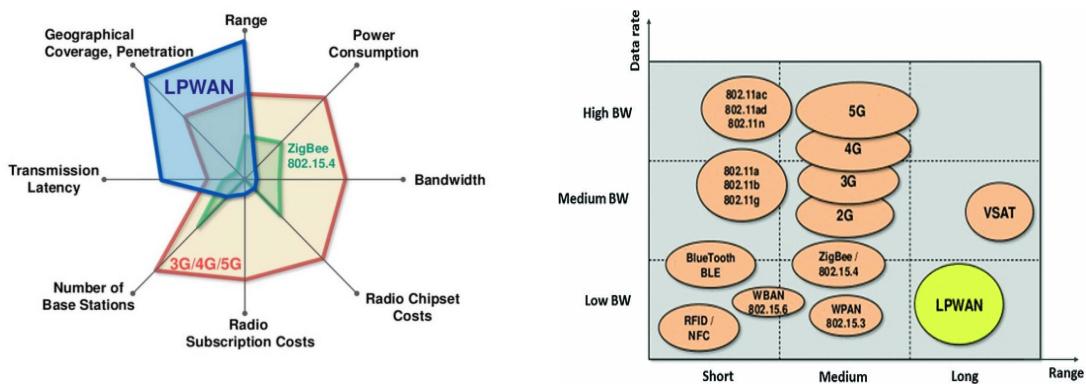
<sup>8</sup>Reliable Communication

<sup>9</sup>Tranceiver

<sup>10</sup>Wireless Sensor Network

<sup>11</sup>Low Rate Wireless Personal Area Network(LR-WPAN)

<sup>12</sup>Zigbee Alliance



(آ) مقایسه بر حسب نرخ داده و بازه پوشش [۶] [۷]  
 (ب) مقایسه بر حسب انرژی، بازه پوشش، نفوذ، تاخیر، هزینه و پهنای باند [۸]

شکل ۳-۲: مقایسه پروتکل زیگبی و سایر پروتکل‌های اینترنت اشیاء

لایه‌های شبکه، امنیت و برنامه و IEEE802.15.4 لایه‌های سختافزار و کنترل دسترسی به رسانه<sup>۱۳</sup> را تعریف می‌کند [۸].

استاندارد شبکه بی‌سیم زیگبی جای خالی‌ای را در بازار پر می‌کند که توسط سایر شرکت‌ها نادیده گرفته شده است. اکثر استانداردهای بی‌سیم به دنبال نزدیک شدن به اینترنت و رسیدن به سرعت‌های بالاتر هستند. اما زیگبی در تلاش است تا با ارائه نرخ ارسال پایین‌تر انتظار مصرف انرژی پایین را برآورد کند. سایر استانداردها به دنبال افزودن ویژگی<sup>۱۴</sup>‌های بیشتر و ارائه خدماتی نظری استریم با کیفیت بالا<sup>۱۵</sup> هستند؛ در حالیکه زیگبی با پشتۀ<sup>۱۶</sup> ای کوچک تنها هدف دارد که بتوان داده‌هایی با حجم کم و تعدد پایین مانند کنترل چراغ یا خواندن داده‌های سنسور دما را ارسال کرد [۹].

## ۲-۳-۲ ویژگی‌های زیگبی

### قابلیت اطمینان بالا

ارتباطات بی‌سیم بطور کلی ارتباطات نامطمئنی هستند اما زیگبی در چند سطح قابلیت اطمینان بالایی فراهم می‌کند که عبارتند از:

<sup>13</sup>Media Access Control(MAC)

<sup>14</sup>Feature

<sup>15</sup>High-Definition Streaming

<sup>16</sup>Stack

- IEEE802.15.4 • در آن از ترکیبی از فناوری‌هایی مانند O-QPSK<sup>۱۷</sup> و DSSS<sup>۱۸</sup> استفاده می‌شود که کارایی بسیار خوبی در محیط‌های با نرخ سیگنال به نویز پایین دارند [۹].
- CSMA-CA : زیگبی از این فناوری برای کنترل دسترسی به رسانه استفاده می‌کند. قبل از ارسال زیگبی به کanal گوش می‌دهد. اگر کسی در حال ارسال نباشد، اطلاعات خود را ارسال می‌کند. این روند از تداخل اطلاعات فرستنده‌های مختلف و ایجاد نیاز برای ارسال دوباره جلوگیری می‌کند [۹].
- کنترل خط: در هر فریم<sup>۲۰</sup> زیگبی از CRC<sup>۲۱</sup> بیتی بعنوان چکسام<sup>۲۲</sup> استفاده می‌شود که قابلیت تشخیص خطای هر فریم را ایجاد می‌کند [۹].
- فریم تایید: هر فریم در کل ۴ بار برای مقصد ارسال می‌شود (۳ بار ارسال مجدد). اگر باز هم فریم نتواند ارسال شود به مبدأ اطلاع داده می‌شود [۹].

### صرف انرژی پایین

ماژول‌های زیگبی در صرف انرژی بسیار بهینه هستند. یک شبکه زیگبی می‌تواند تا چند سال تنها با استفاده از باتری عمل کند. اگر استفاده از زیگبی به درستی مدیریت شود این زمان حتی می‌تواند به عمر باتری اگر بدون استفاده در قفسه بماند برسد. دلیل این استفاده کم این است که گره‌های زیگبی می‌توانند به خواب بروند. نیازی به نگهداشتن ارتباط برای باقی‌ماندن در شبکه ندارند [۹].

### امنیت بالا

زیگبی با استفاده از استاندارد رمزگذاری پیشرفته<sup>۲۳</sup> باعث می‌شود که تنها فرستنده و گیرنده از محتویات فریم اطلاع داشته باشند. همچنین روندی برای احراز هویت گره‌ها هنگام اضافه شدن به شبکه بکار می‌برد که مانع اضافه شدن گره‌های مخرب به شبکه می‌شود [۹].

<sup>17</sup>Offset-Quadrature Phase-Shift Keying

<sup>18</sup>Direct Sequence Spread Spectrum

<sup>19</sup>Carrier Sense Multiple Access Collision Avoidance

<sup>20</sup>Frame

<sup>21</sup>Cyclic Redundancy Check

<sup>22</sup>Checksum

<sup>23</sup>Advanced Encryption Standard(AES)

### ۳-۳-۲ دسته‌بندی دستگاه‌ها

#### دسته‌بندی فیزیکی

با توجه به توانایی‌های پردازشی دو نوع دستگاه در IEEE802.15.4 آورده شده است:

- دستگاه با عملکرد کامل<sup>۲۴</sup>: این دستگاه‌ها توانایی انجام همه‌ی اعمال استاندارد مانند مسیریابی<sup>۲۵</sup>، هماهنگی<sup>۲۶</sup> و ارسال اطلاعات حسگر را دارند. در استاندارد فعلی این دستگاه‌ها باید همیشه روشن باشند [A].
- دستگاه با عملکرد کاهش‌یافته<sup>۲۷</sup>: این دستگاه‌ها تنها توانایی ارسال داده‌های حسگر را دارند و می‌توانند به خواب بروند [A].

#### دسته‌بندی منطقی

در این دسته‌بندی سه نوع دستگاه وجود دارد:

- هماهنگ‌کننده: ریشه درخت شبکه را تشکیل می‌دهد و می‌تواند به شبکه‌های دیگر متصل شود. در هر شبکه دقیقاً یک هماهنگ‌کننده وجود دارد. هماهنگ‌کننده مسئول راه‌اندازی شبکه و انتخاب پارامترهای شبکه مانند کانال فرکانس رادیویی<sup>۲۸</sup>، شناسه یکتای شبکه و تنظیم سایر پارامترهای عملیاتی است. همچنین می‌تواند اطلاعات مربوط به شبکه و کلیدهای امنیتی را ذخیره کند [A].
- مسیریاب: مسیریاب بعنوان گره میانی عمل می‌کند و داده‌ها را از دستگاه‌های دیگر منتقل می‌کند. مسیریاب می‌تواند به یک شبکه از قبل موجود متصل شود، همچنین می‌تواند درخواست اتصال سایر دستگاه‌ها را بپذیرد و نوعی فرستنده مجدد به شبکه باشد [A].

- گره پایانی: این دستگاه می‌تواند دستگاه‌های کم‌صرف یا با باتری باشد. آنها می‌توانند اطلاعات مختلفی را از حسگرها جمع‌آوری کنند و عملکرد کافی برای صحبت با والدین خود(همماهنگ‌کننده یا مسیریاب) را دارند اما نمی‌توانند داده‌ها را از دستگاه‌های دیگر ارسال کنند. این عملکرد

<sup>24</sup>Fully Functional Device(FFD)

<sup>25</sup>Routing

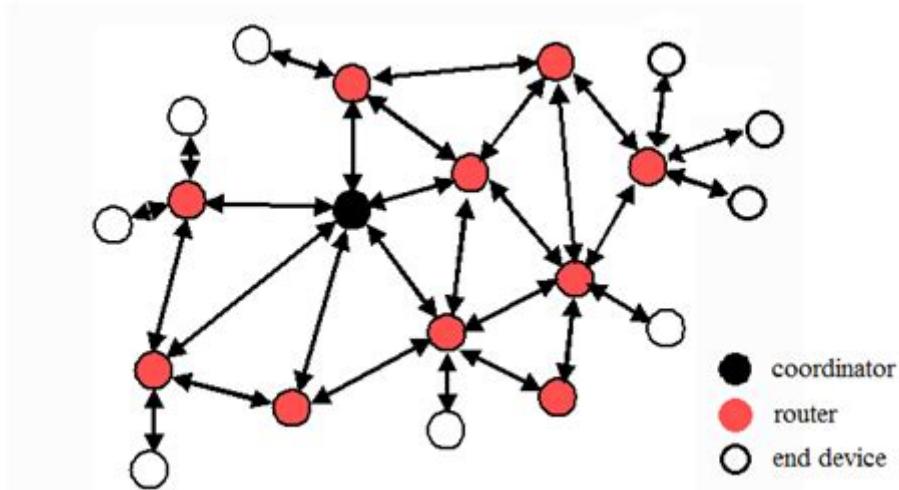
<sup>26</sup>Coordination

<sup>27</sup>Reduced Functional Device(RFD)

<sup>28</sup>Radio Frequency Channel

کاهش یافته امکان کاهش هزینه را فراهم می‌کند. این دستگاه‌ها مجبور نیستند تمام مدت بیدار بمانند، در حالیکه دستگاه‌های متعلق به دو دسته دیگر باید بیدار بمانند [۱۰].

در شکل ۴-۲ [۱۰] می‌توان یک شبکه زیگبی را که شامل یک هماهنگ‌کننده، چند مسیریاب و چند گره پایانی است را مشاهده کرد.



شکل ۴-۲: شبکه زیگبی شامل هماهنگ‌کننده، مسیریاب و گره پایانی [۱۰]

### ۴-۳-۲ توپولوژی

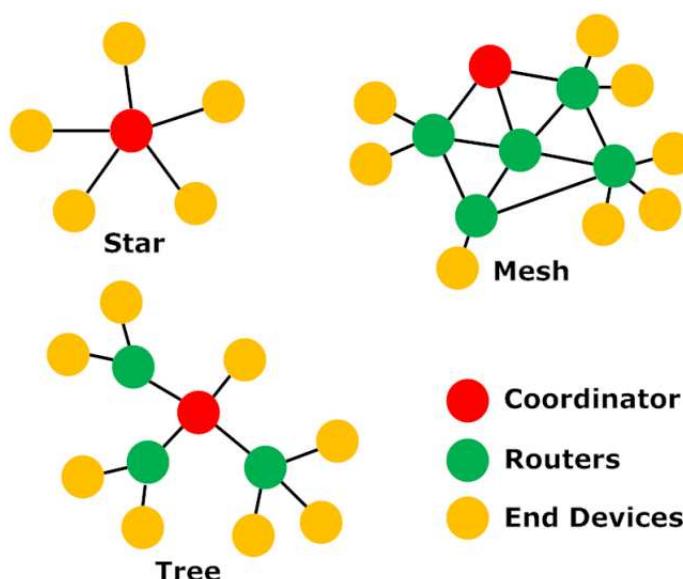
در زیگبی می‌توان سه توپولوژی زیر را داشت:

- ستاره: توپولوژی ستاره از یک هماهنگ‌کننده و تعدادی دستگاه پایانی تشکیل شده است. در توپولوژی ستاره، یک مدل شبکه master slave اتخاذ می‌شود که در آن هماهنگ‌کننده و دستگاهی با عملکرد کامل است و دستگاهی با عملکرد کامل یا کاهش یافته است. دستگاه‌های انتهایی زیگبی بصورت فیزیکی و الکتریکی از یکدیگر جدا می‌شوند و اطلاعات را تنها از طریق هماهنگ‌کننده منتقل می‌کنند [۱۰].

- درخت خوش‌های<sup>۲۹</sup>: این توپولوژی مانند توپولوژی ستاره است، با این تفاوت که در این توپولوژی سایر دستگاه‌ها می‌توانند با یکدیگر ارتباط برقرار کنند تا دستگاه‌های بیشتری بتوانند به دستگاه‌های با عملکرد کامل و غیر هماهنگ‌کننده متصل شوند تا بتوان شبکه را از لحاظ جغرافیایی گسترش داد [۱۰].

<sup>29</sup>Cluster Tree

- مش<sup>۳۰</sup>: در توپولوژی مش، هر گره می‌تواند با هر گره دیگری در محدوده خود ارتباط برقرار کند. توپولوژی مش برای نگهداری پیچیده است اما نسبت به خطأ مقاوم‌تر و قابل تحمل‌تر است [۱۸]. در شکل ۵-۲ [۱۹] می‌توان نقش گره‌های مختلف در توپولوژی‌های ذکر شده را مشاهده کرد.



شکل ۲-۵: توپولوژی‌های مختلف شبکه زیگبی [۱۹]

### ۵-۳-۲ دیجی بین‌الملل

دیجی بین‌الملل<sup>۳۱</sup> نام یکی از تولیدکنندگان ماژول‌های زیگبی است. ماژولی که ما در این پژوه استفاده می‌کنیم تولید این شرکت است. مدل این ماژول Digi XBee S2 است که می‌توان تصویر آن را در شکل ۶-۲ مشاهده کرد.

### ۶-۳-۲ طرز کار دستگاه‌های ایکس‌بی

دستگاه‌های ایکس‌بی از طریق هوا با یکدیگر ارتباط برقرار می‌کنند و پیام‌ها را به‌شکل بی‌سیم ارسال و دریافت می‌کنند. دستگاه‌ها فقط می‌توانند پیام‌ها را منتقل کنند ولی نمی‌توانند آنها را مدیریت کنند. با این حال، ماژول‌ها می‌توانند با دستگاه‌های هوشمند از طریق رابط سریال ارتباط برقرار کنند. دستگاه‌های ایکس‌بی داده‌های ورودی سریال را از طریق هوا منتقل می‌کنند و هر چیزی را که بصورت بی‌سیم دریافت می‌شود به خروجی سریال می‌فرستند. چه برای مقاصد ارتباطی و چه صرفاً برای

<sup>30</sup>Mesh

<sup>31</sup>Digi International



شکل ۲-۶: ماژول زیگبی مدل XBee S2

پیکربندی دستگاه، ترکیبی از هر دو فرایند برای برقراری ارتباط لازم است. به این صورت، دستگاه‌های هوشمند مانند میکروکنترلر<sup>۳۲</sup>‌ها یا رایانه‌های شخصی می‌توانند آنچه را که دستگاه ارسال می‌کند کنترل کرده و پیام‌های دریافتی و ارسالی را مدیریت کنند [۱۲]. دو فرایند لازم برای ارتباط را می‌توان در شکل ۷-۲ [۱۲] دید که عبارتند از:

- ارتباط بی‌سیم: این ارتباط بین ماژول‌های ایکس‌بی انجام می‌شود. ماژول‌هایی که قرار است با هم کار کنند باید بخشی از یک شبکه باشند و باید از یک فرکانس رادیویی استفاده کنند. همه ماژول‌هایی که این الزامات را برآورده می‌کنند می‌توانند بصورت بی‌سیم با یکدیگر ارتباط برقرار کنند.
- ارتباط سریال: این ارتباط بین ماژول ایکس‌بی و دستگاه هوشمند متصل به آن از طریق رابط سریال صورت می‌گیرد.



شکل ۲-۷: فرایندهای لازم برای ارتباط دو ماژول ایکس‌بی [۱۲]

<sup>32</sup>Microcontroller

## ۷-۳-۲ ارتباط سریال ماژول‌های ایکس‌بی

دستگاه‌های ایکس‌بی می‌توانند از اتصال سریال محلی خود به روش‌های بسیار متفاوتی استفاده کنند. حالت عملیاتی نحوه ارتباط دستگاه میزبان با ماژول ایکس‌بی را از طریق رابط سریال تعیین می‌کند. این حالت‌ها به دو دسته تقسیم می‌شوند که در ادامه به شرح هر کدام می‌پردازیم.

### حالت شفاف

هنگام کار در حالت شفاف<sup>۳۳</sup>، یک ماژول ایکس‌بی بعنوان جایگزین خط سریال عمل می‌کند. تمام داده‌های دریافت شده از طریق ورودی سریال بلافصله از طریق هوا منتقل می‌شود. هنگامی که ماژول ایکس‌بی داده‌های بی‌سیم را دریافت می‌کند، از طریق رابط سریال به همان شکل محل دریافت شده ارسال می‌شود. در واقع، ارتباط در حالت شفاف مانند آن است که دو ماژول توسط یک سیم به هم متصل شده باشند<sup>[۱۲]</sup>.

برای ارتباط دو ماژول ایکس‌بی، ماژول ارسال‌کننده به آدرس گیرنده نیاز دارد. هنگام کار در حالت شفاف، باید این آدرس را در ماژولی که در حال ارتباط است پیکربندی کرد. ماژول‌های ایکس‌بی می‌توانند آدرس ۶۴ بیتی کامل ماژول مقصد را ذخیره کنند<sup>[۱۲]</sup>. این آدرس باید در دو پارامتر نوشته شود: آدرس مقصد بالا<sup>۳۴</sup> و آدرس مقصد پایین<sup>۳۵</sup>. بعنوان مثال برای ارتباط دو ماژول در این حالت باید مثل شکل ۸-۲ [۱۲] دو ماژول را پیکربندی کرد.



شکل ۸-۲: ارتباط سریال در حالت شفاف [۱۲]

حالت شفاف محدودیت‌هایی دارد. بعنوان مثال، هنگام کار با چند ماژول، باید قبل از ارسال هر پیام، مقصد را پیکربندی کرد. با این حال، حالت شفاف به دلایل زیر راهی آسان برای شروع کار با دستگاه‌های ایکس‌بی فراهم می‌کند:

<sup>33</sup>Transparent Mode

<sup>34</sup>Destination Address High(DH)

<sup>35</sup>Destination Address Low(DL)

- عملیات بسیار ساده است.
- آنچه ارسال می‌شود دقیقاً همان چیزی است که در مازول دیگر دریافت می‌شود.
- سازگار با هر دستگاهی است که بتواند از طریق یک رابط سریال ارتباط برقرار کند.

### حالت API

حالت API یک رابط ساختاریافته را فراهم می‌کند که در آن داده‌ها از طریق رابط سریال در بسته‌های سازماندهی شده و به ترتیب تعیین شده ارتباط برقرار می‌کنند. این روند اجازه می‌دهد بدون نیاز به تعریف پروتکل، ارتباط پیچیده بین دستگاه‌ها برقرار کرد [۱۲].

در حالت شفاف تمام داده‌های دریافت شده از طریق ورودی سریال برای ارسال رادیویی در صورت قرار می‌گیرند و داده‌های دریافت شده به صورت بی‌سیم دقیقاً همان‌طور که دریافت شده‌اند، بدون اطلاعات اضافی به خروجی سریال ارسال می‌شوند [۱۲].

به دلیل این رفتار، دستگاه‌هایی که در حالت شفاف کار می‌کنند محدودیت‌هایی دارند:

- برای خواندن یا نوشتن پیکربندی یک دستگاه در حالت شفاف، ابتدا باید دستگاه را به حالت فرمان تغییر داد.
- اگر دستگاهی نیاز به انتقال پیام به دستگاه‌های مختلف داشته باشد، باید پیکربندی آن برای ایجاد یک مقصد جدید بروز شود. دستگاه باید برای تنظیم مقصد وارد حالت فرمان شود.
- دستگاهی که در حالت شفاف کار می‌کند نمی‌تواند منبع پیام بی‌سیمی را که دریافت کرده شناسایی کند. اگر نیاز به تمایز بین داده‌های دریافتی از دستگاه‌های مختلف باشد، دستگاه‌های فرستنده باید شامل اطلاعات اضافی شناخته شده توسط همه دستگاه‌ها باشند تا بتوان بعداً آنها را استخراج کرد.

حالت API راه بسیار آسان‌تری برای انجام اقدامات ذکر شده ارائه می‌دهد:

- از آنجایی که فریم‌های مختلفی برای اهداف مختلف (مانند پیکربندی و ارتباط) وجود دارد، می‌توان یک دستگاه را بدون وارد شدن به حالت فرمان پیکربندی کرد.
- از آنجایی که مقصد داده بعنوان بخشی از ساختار فریم API گنجانده شده است، می‌توان از حالت API برای انتقال پیام‌ها به چندین دستگاه استفاده کرد.

- فریم API همانطور که در شکل ۹-۲ [۱۲] نشان داده شده است، شامل منبع پیام است. بنابراین تشخیص اینکه داده‌ها از کجا آمده‌اند آسان است.



شکل ۹-۲: ارتباط سریال در حالت API [۱۲]

## ۴-۲ دروازه

برای دریافت، تجمعیع، قراردادن داده‌ها در قالب مخصوص و ارسال به سرور به یک دستگاه بعنوان دروازه در هر کارخانه نیاز داریم. با توجه به اینکه ممکن است در یک لحظه چند گره انتهایی اطلاعات خود را برای دروازه ارسال کنند و با توجه به امکان نباز به انجام چند کار بصورت همزمان، نیاز به سخت‌افزاری با قابلیت چندرشته‌ای<sup>۳۶</sup> و منابع کافی برای جمع‌آوری و ارسال داده‌ها داریم. یکی از انتخاب‌های مناسب برای این کار رایانه‌های تک-بورد رزبری‌پای<sup>۳۷</sup> است.

مدل استفاده شده در این پروژه رزبری‌پای ۴ مدل بی<sup>۳۸</sup> است که تصویر آن در شکل ۱۰-۲ نشان داده شده است. رزبری‌پای ۴ مدل بی جدیدترین محصول از سری کامپیوتراهای محبوب رزبری‌پای است. در مقایسه با نسل قبلی، سرعت پردازنده، عملکرد چندسنه‌ای، حافظه و اتصال افزایش یافته است، در حالیکه سازگاری با محولات قبلی و مصرف انرژی مشابه حفظ شده است. برای کاربر، این مدل عملکرد قابل مقایسه‌ای با سیستم‌های رایانه شخصی x86 را ارائه می‌دهد. از ویژگی‌های کلیدی این محصول می‌توان به پردازنده چهار هسته‌ای ۶۴ بیتی با کارایی بالا، پشتیبانی از نمایشگر دوگانه با رزولوشن تا 4K از طریق یک جفت پورت micro-HDMI، رم تا ۴ گیگابایت، LAN بی‌سیم ۲/۴ و ۵ گیگاهرتز دو بانده، بلوتوث نسخه ۵ و درگاه USB 3.0 اشاره کرد[۱۳].

<sup>36</sup>Multi Threading

<sup>37</sup>Raspberry Pi Single-Board Computer

<sup>38</sup>Raspberry Pi 4 Model B



شکل ۲: بورد زبری‌پایی ۴ مدل بی

## ۵-۲ پایگاه داده‌ی سری زمانی

سری زمانی یک توالی مرتب شده از مقادیر یک متغیر در فواصل زمانی مساوی است. بنابراین سری زمانی یک توالی از داده‌های زمان گستته است [۱۴]. داده‌های لرزش جمع‌آوری شده در این پروژه نیز از نوع سری زمانی هستند. جهت ذخیره‌سازی، پردازش و مدیریت بهینه این داده‌ها نیاز به یک پایگاه داده داریم.

یک پایگاه داده سری زمانی بدون طرحواره<sup>۳۹</sup> منبع باز با اجزای اختیاری منبع بسته است که توسط InfluxData توسعه یافته است. این به زبان برنامه‌نویسی گو<sup>۴۰</sup> نوشته شده است و برای مدیریت داده‌های سری زمانی بهینه شده است. این یک زبان پرس و جو<sup>۴۱</sup> مانند SQL را ارائه می‌دهد. نسخه منبع باز که در شکل ۱۱-۲ [۱۵] نشان داده شده است، یک پلتفرم پایگاه داده سری زمانی کامل را با خدمات مختلف از جمله هسته InfluxDB فراهم می‌کند و می‌تواند بر روی ابر و در محل روی یک گره واحد اجرا شود.

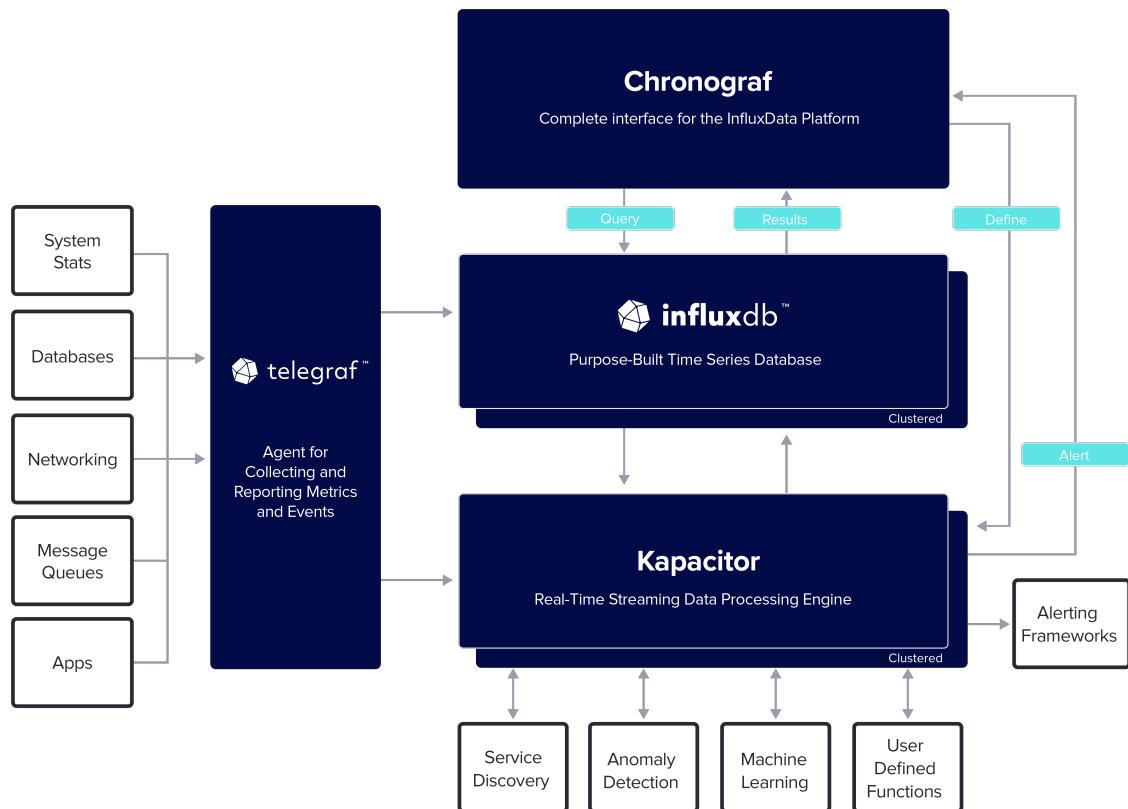
استفاده از پایگاه داده InfluxDB برتری‌هایی نسبت به سایر پایگاه داده‌ها دارد که باعث شد ما از آن برای ذخیره داده‌های لرزش استفاده کنیم. این برتری‌ها عبارتند از:

- طراحی اختصاصی برای داده‌های سری زمانی و عملکرد بهینه در خواندن و نوشتمن این داده‌ها.
- افزودنی‌های رایگان و متعدد مانند HTTP API که باعث تسریع و راحتی توسعه می‌شود.

<sup>39</sup>Schemaless

<sup>40</sup>Gō

<sup>41</sup>Query Language



[۱۵] شکل ۲: پلتفرم متن باز پایگاه داده سری زمانی InfluxDB

- پشتیبانی از تعداد بالای زبان‌های برنامه‌نویسی.
- پشتیبانی از زبان پرس و جویی مانند SQL بنام InfluxQL که یادگیری و پیاده‌سازی را راحت‌تر می‌کند.

## ۶-۳ پیش‌پردازش

داده‌های جمع‌آوری شده در دروازه قبل از تحویل به مدل هوش مصنوعی نیاز به برخی پیش‌پردازش‌ها دارند. زبان برنامه‌نویسی انتخابی ما پایتون<sup>۴۲</sup> است. این زبان به دلیل راحتی در توسعه و وجود کتابخانه‌های زیاد در حوزه هوش مصنوعی انتخاب شده است.

کتابخانه‌ی استفاده شده برای پیش‌پردازش داده‌ها نامپای<sup>۴۳</sup> نام دارد. نامپای یک کتابخانه منبع باز<sup>۴۴</sup> توسعه‌یافته توسط جامعه<sup>۴۵</sup> است که یک شیء آرایه پایتون چندبعدی را به همراه توابع مرتبط با آرایه که

<sup>42</sup>Python

<sup>43</sup>NumPy

<sup>44</sup>Open Source

<sup>45</sup>Community-Developed

روی آن کار می‌کنند ارائه می‌دهد. نامپای در پردازش عددی از طریق ndarray چندبعدی تخصص دارد، جایی که آرایه‌ها اجازه عملیات عنصر به عنصر<sup>۴۶</sup> یا پخش<sup>۴۷</sup> را می‌دهند. در صورت نیاز، جبر خطی را می‌توان بدون از قبل تغییر دادن آرایه‌های نامپای استفاده کرد. علاوه بر این، اندازه آرایه‌ها را می‌توان بصورت پویا تغییر داد. این مسئله نگرانی‌هایی را که معمولاً برنامه‌نویسی سریع در زبان‌های دیگر را سودگم می‌کند، از بین می‌برد<sup>[۱۶، ۱۷]</sup>.

## ۷-۲ برنامه‌ی تحت وب

پس از دریافت داده‌های جمع‌آوری شده در دروازه و انجام مراحل پیش‌پردازش لازم، اکنون داده‌ها در اختیار مدل یادگیری ماشینی قرار می‌گیرد تا طول عمر مفید باقیمانده‌ی تجهیزات را پیش‌بینی کند. ما برنامه‌ی وبی را برای نمایش به متخصصان پیاده‌سازی کرده‌ایم تا بتوانند تعمیر و نگهداری بعدی خود را طبق پیش‌بینی برنامه‌ریزی کنند. برای پیاده‌سازی برنامه‌ی تحت وب ما از چارچوب ویو<sup>۴۸</sup> استفاده کرده‌ایم.

ویو یک چارچوب مترقبی<sup>۴۹</sup> برای ساخت رابطه‌ای کاربری است. برخلاف سایر چارچوب‌های یکپارچه، ویو از ابتدا به گونه‌ای طراحی شده است که برای رویکرد افزایشی قابل‌پذیرش باشد. هسته‌ی کتابخانه فقط بر روی لایه‌ی نما<sup>۵۰</sup> مرکز است و بر احتی قابل‌انتخاب و ادغام با کتابخانه‌های دیگر یا پروژه‌های موجود است. از سوی دیگر، ویو قادر است برنامه‌های پیچیده تک صفحه‌ای را در ترکیب با ابزارهای نوین و کتابخانه‌های پشتیبان ایجاد کند<sup>[۱۸]</sup>.

ویو دو ویژگی اصلی دارد:

- تفسیر اعلامی<sup>۵۱</sup>: ویو HTML استاندارد را با استفاده از نحو template گسترش می‌دهد که امکان توصیف صریح خروجی HTML بر اساس وضعیت جاوا اسکریپت<sup>۵۲</sup> را فراهم می‌کند.

- واکنش‌پذیری<sup>۵۳</sup>: ویو بطور خودکار وضعیت جاوا اسکریپت را دنبال می‌کند و در صورت تغییر،

<sup>46</sup>Element by Element

<sup>47</sup>Broadcast

<sup>48</sup>Vue Framework

<sup>49</sup>Progressive

<sup>50</sup>View Layer

<sup>51</sup>Declarative Rendering

<sup>52</sup>JavaScript

<sup>53</sup>Reactivity

مدل سند شیء<sup>۵۴</sup> را بروزرسانی می‌کند.

ویو چارچوب و اکوسیستمی است که اکثر ویژگی‌های مشترک مورد نیاز در توسعه سمت کاربر را پوشش می‌دهد. اما وب بسیار متنوع است. چیزهایی که در وب می‌سازیم ممکن است از نظر شکل و مقیاس بسیار متفاوت باشند. با توجه به این موضوع، ویو به‌شکلی طراحی شده است که منعطف باشد. اما با وجود انعطاف‌پذیری دانش اصلی در مورد نحوه عملکرد ویو در همه کاربردها یکسان است[۱۸]. این ویژگی‌ها چارچوب ویو را به چارچوبی کاربردی، کم حجم و آسان برای یادگیری تبدیل می‌کند.

## ۸-۲ جمع‌بندی و نتیجه‌گیری

در این بخش، به تکنولوژی‌ها، چارچوب‌ها و قطعات استفاده شده در پروژه پرداختیم. هریک از آنها را معرفی کرده و سپس ویژگی‌ها و دلیل انتخابشان را بیان کردیم. ابتدا درباره حسگر استفاده شده برای اندازه‌گیری لرزش صحبت کردیم. سپس به سراغ گرههای انتهایی و بورد آردوینوی انتخاب شده رفتیم. بعد از آن پروتکل ارتباطی زیگبی، ویژگی‌ها، طرز کار و ماژول ایکس‌بی را معرفی کردیم. سپس دروازه و بورد رزبری‌پای استفاده شده را شناختیم. پایگاه داده‌ی InfluxDB که مناسب داده‌های سری زمانی بود را معرفی کردیم. برای پیش‌پردازش داده‌ها از کتابخانه نامپای گفتیم و در انتهای نیز چارچوب ویو که برای پیاده‌سازی برنامه تحت وب استفاده شده بود را مرور کردیم.

<sup>54</sup>Document Object Model

# فصل سوم

## پیاده‌سازی سیستم

در این قسمت نحوه اجرای پروژه را بطور کامل توضیح خواهیم داد. در هر قسمت ساختار فایل‌ها، نحوه پیاده‌سازی و چالش‌هایی که با آن مواجه بودیم بهمراه نحوه حل آنها را توضیح خواهیم داد.

## ۱-۳ معماری کلی سیستم

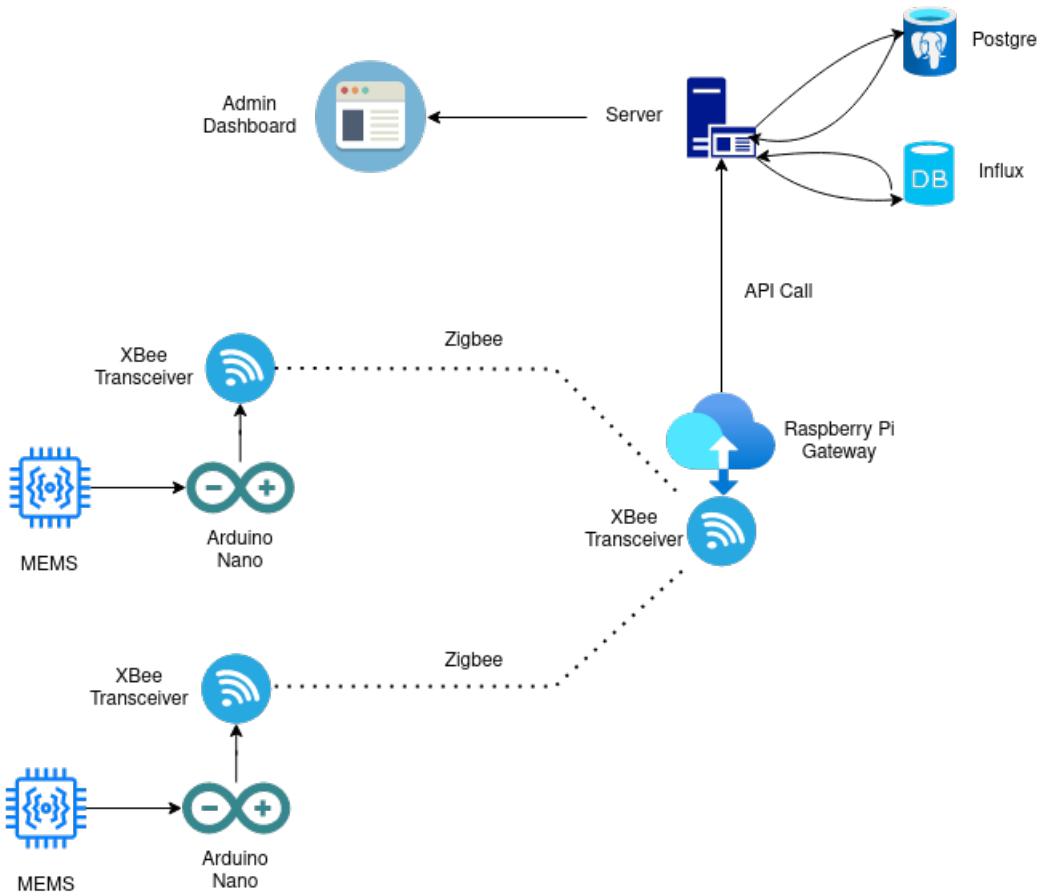
برای درک بهتر این پروژه، نمایی کلی از سیستم و نحوه عملکرد آن را همانطور که در [شکل ۱-۳](#) نشان داده شده است، ارائه خواهیم داد. این پروژه در چندین محل صنعتی مستقر خواهد شد. در هر کارخانه چندین میکروکنترل آردوینو، با یک سنسور لرزش و یک فرستنده گیرنده زیگبی متصل، روی هر قطعه از تجهیزات نصب شده است. داده‌ها در زمان‌های برنامه‌ریزی شده حس می‌شوند و سپس به دروازه‌ای در کارخانه ارسال می‌شوند. دروازه اطلاعاتی را که از فرستنده‌گیرنده‌های زیگبی می‌آید را جمع‌آوری می‌کند تا زمانی که به یک آستانه‌ی از پیش تعریف شده برسد. سپس دروازه با سرور مرکزی احراز هویت می‌شود و سپس داده‌های جمع‌آوری شده را با پروتکل HTTP به سرور ارسال می‌کند. در سرور مراحل پیش‌پردازش لازم انجام می‌شود و سپس داده‌ها در پایگاه داده‌ی سری زمانی ذخیره می‌شوند. سپس مدل یادگیری ماشین با داده‌های قبلی و جدید اجرا می‌شود تا پیش‌بینی دقیق‌تری درباره طول عمر مفید باقیمانده تجهیزات به ما ارائه دهد. در صورت درخواست، تمام داده‌های تحلیلی و پیش‌بینی‌ها از طریق داشبورد مدیریت قابل دسترسی است. متخصصان می‌توانند از این داده‌ها برای برنامه‌ریزی زمان نگهداری مناسب استفاده کنند.

## ۲-۳ گره انتهایی و آردوینو

برای پیاده‌سازی گره انتهایی، سه بخش را باید پیاده‌سازی می‌کردیم که عبارتند از:

- خواندن از حسگر و نمونه‌برداری
- تشکیل فریم زیگبی و نوشتن در درگاه سریال
- خوابیدن و بیدارشدن در بازه‌های زمانی مشخص

در ادامه هر بخش را توضیح می‌دهیم.



شکل ۳-۱: نمای کلی سیستم

### ۱-۲-۳ خواندن از حسگر و نمونه‌برداری

حسگر ADXL345 چندین پارامتر دارد که باید در زمان راهاندازی تنظیم شوند. برای انجام پیکربندی و همچنین خواندن داده‌های حسگرها بدون نیاز به استفاده مستقیم از پروتکل I2C، از کتابخانه<sup>۱</sup> استفاده کردہ‌ایم. یکی از پارامترها محدوده شتاب است که با تابع `setRange` تنظیم می‌شود. محدوده مجاز ۲، ۴، ۸ یا ۱۶ برابر شتاب گرانش زمین است. اما هرچه دامنه بزرگتر شود، داده‌ها دقیق‌تر کمتری پیدا می‌کنند. از آنجایی که یک قطعه از تجهیزات حتی زمانی که قدیمی باشد با شتاب بالا نمی‌لرزد و برای پیش‌بینی‌های دقیق‌تر به دقت نیاز است، تصمیم گرفتیم که محدوده را روی دو برابر شتاب گرانش زمین تنظیم کنیم.

پارامتر دیگر نرخ داده است که باید در زمان راهاندازی با استفاده از تابع `setDataRate` تنظیم شود. نرخ داده برای تنظیم نرخ ارتباط بین حسگر و بورد آردوینو از طریق پروتکل I2C استفاده می‌شود. بنابراین مهم است که نرخ داده با رسانه‌ی ارتباطی سازگار باشد. از آنجایی که اگر نرخ داده بالاتر از نرخی

<sup>1</sup> Adafruit\_ADXL345\_U.h

باشد که رسانه قادر به انجام آن است، برخی از داده‌ها از بین می‌روند. بدلیل برخی چالش‌های دیگر که در ادامه در مورد آنها صحبت خواهد شد، نرخ را روی ۱۰۰ هرتز تنظیم کرده‌ایم. برای نمونه‌برداری از داده‌ها در فرکانس نمونه‌برداری مورد نظر، تأخیر نمونه‌برداری برای تأخیر بین هر نمونه حسگر محاسبه می‌شود.

برای هر نمونه، از یک رویداد برای بدستآوردن شتاب در سه محور x, y, z استفاده می‌کنیم. سپس داده‌ها در یک آرایه ذخیره می‌شوند تا بعداً قالب‌بندی و با پروتکل زیگبی ارسال شوند. اگر فرمت و ارسال در حین نمونه‌برداری انجام شود، بدلیل تأخیر در فرمت، تشکیل و ارسال فریم‌ها از طریق درگاه سریال، فرکانس نمونه‌برداری صحیح نخواهد بود.

ما چالش دیگری در این شرایط بوجود می‌آید. با توجه به محدودیت حافظه موجود در دستگاه آردوینو نانو، فضای ایجاد آرایه‌ها و ذخیره داده‌ها نیز محدود است. به همین دلیل است که نمی‌توانیم به فرکانس‌های نمونه‌برداری بالا دست پیدا کنیم.

### ۲-۲-۳ تشکیل فریم زیگبی و نوشتن در درگاه سریال

همانطور که در بخش ۷-۳-۲ گفته شد، در حالت شفاف پیام‌ها به‌شکل مستقیم منتقل می‌شوند اما در حالت API نیاز به تشکیل فریم دارند. ابتدا سراغ استفاده از کتابخانه‌های آماده برای تشکیل و نوشتن فریم بر روی درگاه سریال رفتیم. یکی از آنها کتابخانه xbee-arduino<sup>۲</sup> بود. اما این کتابخانه مشکلات فراوانی داشت که سبب عدم ارسال فریم توسط ماژول می‌شد. یکی از این مشکلات محاسبه اشتباه چک‌سام و عدم مطابقت با استاندارد بود. مشکل دیگر نیز در نوشتن بر روی درگاه سریال بود. همه این مشکلات باعث شد تا ما کتابخانه خود را پیاده‌سازی کنیم. اگرچه این پیاده‌سازی کلی نیست و بعضی برای این پروژه انجام شده است، اما کارکرد مورد انتظار را دارد.

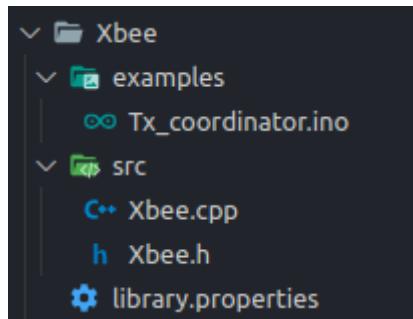
#### Xbee کتابخانه

ساختار فایل کتابخانه ایکس‌بی در شکل ۳-۲ نشان داده شده است. ساختار طبق استانداردهای آردوینو<sup>۳</sup> برای ایجاد یک کتابخانه است. فایل‌های اصلی در پوشه src قرار دارند، یک فایل هدر که توابع، ورودی‌ها و خروجی‌ها را تعریف می‌کند و یک فایل سی‌پلاس‌پلاس برای پیاده‌سازی و تکمیل آن توابع. در پوشه نمونه‌ها، فایلی با فرمت ino وجود دارد که کاربرد اصلی کتابخانه برای انتقال یک فریم در حالت API را

<sup>2</sup><https://github.com/andrewrapp/xbee-arduino>

<sup>3</sup><https://arduino.github.io/arduino-cli/0.33/library-specification/>

نشان می‌دهد. یک فایل library.properties در ریشه پوشه وجود دارد. این فایل برای تشریح برخی از ویژگی‌های این کتابخانه از جمله نام، نویسنده، توضیحات، نسخه کتابخانه، معماری‌های سازگار و اینکه این کتابخانه به کدام کتابخانه‌ها وابستگی دارد استفاده می‌شود. این فایل به IDE آردوینو کمک می‌کند تا کتابخانه و وابستگی‌های آن را نصب کند.



شکل ۳-۲: ساختار فایل کتابخانه ایکس بی

### ساختار فریم API

ساختار کلی فریم API در شکل ۳-۳ [۱۲] نشان داده شده است. فریم با یک بایت جداکننده شروع می‌شود، سپس با ۲ بایت برای نشان‌دادن طول فریم ادامه می‌یابد و با یک بایت چکسام پایان می‌یابد. بایتهای بین طول و فیلد چکسام را داده‌های فریم می‌نامند که برای هر نوع بسته API متفاوت است. بایت اول فیلد طول مهم‌ترین بایت<sup>۴</sup> و بایت بعدی کم‌همیت‌ترین<sup>۵</sup> است. اولین بایت از داده‌های فریم نشان‌دهنده نوع فریم است. چکسام برای کمک به تست یکپارچگی داده‌ها استفاده می‌شود. اگر مقدار چکسام به اشتباه محاسبه شده باشد یا با چکسام واقعی داده‌های درون فریم متفاوت باشد، فریم کنار گذاشته و نادیده گرفته می‌شود.

Start delimiter	Length		Frame type	Frame data										Checksum
				5	6	7	8	9	...	n				
1	2	3	4											n+1
0x7E	MSB	LSB	API frame type	Frame-type-specific data										Single byte

شکل ۳-۳: ساختار کلی فریم API [۱۲]

روش صحیح محاسبه چکسام فریم به شرح زیر است:

<sup>4</sup>Most Significant Byte(MSB)

<sup>5</sup>Least Significant Byte(LSB)

۱. تمام بایت‌های بسته به استثنای محدود‌کننده شروع و طول با هم جمع می‌شوند.

۲. از نتیجه فقط پایین‌ترین ۸ بیت نگه داشته می‌شود.

۳. مقدار بدست‌آمده از  $0xFF$  کم می‌شود.

## فریم درخواست ارسال

فریم‌های API بسته به عملکردشان از چندین نوع فریم پشتیبانی می‌کنند. در این بخش فقط در مورد فریم درخواست ارسال که برای ما مهم است صحبت خواهیم کرد. بخش‌های مختلف داده‌های فریم درخواست ارسال در [جدول ۱-۳](#) به همراه کاربرد آمده است.

### داده ارسالی

برای استفاده بهینه از فضای پیام در هر فریم، تصمیم گرفتیم ارتعاشات حسگر را به صورت زیر ارسال کنیم:

- هر داده با دقت ۳ رقم اعشار ارسال می‌شود.
- برای جداسازی ارتعاش بر اساس محور در پیام ارسالی، از حروف x، y و z استفاده نمی‌کنیم.
- اولین بایت همیشه شناسه اندازه‌گیری را نشان می‌دهد. پس از آن پنج بایت اول نشان‌دهنده محور ایکس، پنج بایت بعدی نشان‌دهنده محور وای و پنج بایت بعدی نشان‌دهنده محور زد است. هر داده در پنج بایت نمایش داده می‌شود، برای مثال عدد  $\frac{452}{3}$ - بصورت {’2,’3,’4,’5,’-} نمایش داده می‌شود.

در سمت گیرنده، داده‌ها باید به حالت اولیه تبدیل شوند.

## فایل Xbee.cpp و توابع مهم

برای تنظیم آدرس مقصد برای هر فریم کلاس XbeeDestAddress را تعریف کردیم. این کلاس شامل فیلد ۶۴ بیتی و ۱۶ بیتی برای آدرس است. با توجه به اینکه به آرایه‌ای از بایت‌ها برای تشکیل فریم نیاز داریم، توابعی متناظر برای گرفتن هر یک از فیلدها نیز نوشته شده است.

برای تشکیل فریم درخواست انتقال کلاس XbeeRequest تعریف شده است. تابع constructFrame برای ایجاد فریم با استفاده از شناسه فریم و داده مورد نظر نوشته شده است. این تابع فریم را طبق

جدول ۳-۱: بخش‌های مختلف داده‌های فریم درخواست ارسال [۱۲]

نام بخش	شماره بایت	نمونه	توضیحات
نوع فریم	۳	0x10	بایت ۰x10 نشان‌دهنده فریم درخواست ارسال است.
شماره فریم	۴	0x01	فریم داده را به گیرنده برای ارسال فریم وضعیت ارسال می‌شناساند. تنظیم کردن با مقدار پاسخ را غیرفعال می‌کند.
آدرس ۶۴ بیتی مقصد	۵ تا ۱۲	0x0000000000000000	آدرس ۶۴ بیتی مقصد را مشخص می‌کند. آدرس مقاصد خاص: هماهنگ کننده: 0x0000000000000000 همه‌پخشی: 0x000000000000FFFF نامعلوم(اگر آدرس مقصد را ندانیم): 0xFFFFFFFxFFFFFFF
آدرس ۱۶ بیتی مقصد	۵ تا ۱۲	0x0000	آدرس ۱۶ بیتی مقصد را مشخص می‌کند. آدرس مقاصد خاص: هماهنگ کننده: 0x0000 نامعلوم یا همه‌پخشی: 0xFFFF
شعاع پخش	۱۵	0x00	حداکثر تعداد پرش‌هایی که انتقال همه‌پخشی می‌تواند پخش شود را تنظیم می‌کند. اگر روی تنظیم شود، شعاع پخش روی حداکثر مقدار پرش تنظیم می‌شود.
انتخاب‌ها	۱۶	0x00	گزینه‌های انتقال پشتیبانی شده: غیرفعال کردن تلاش ارسال مجدد - 0x01 فعال کردن رمزگذاری APS (با این کار حداکثر بایت‌های داده ارسالی تا ۴ بایت کاهش می‌یابد) 0x20 - استفاده از مهلت زمان طولانی برای ارسال - 0x40
داده ارسالی	۱۷ به بعد	0x...	تا ۲۵۵ بایت داده ارسالی برای مقصد

بخش ۳-۲-۳ ایجاد کرده و فریم تشکیل شده برگردانده می‌شود. با استفاده از تابع `writeFrameToSerial`

نیز می‌توان فریم تشکیل شده را بر روی درگاه سریال دلخواه نوشت.

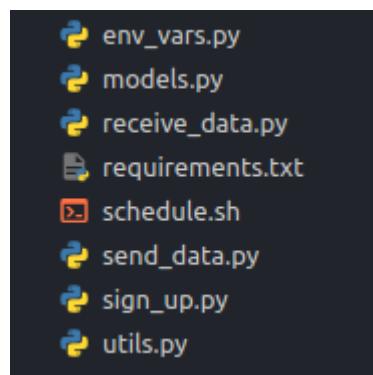
### ۳-۲-۳ خوابیدن و بیدارشدن در بازه‌های زمانی مشخص

برای کاهش مصرف انرژی، حسگر تنها در زمان‌های مشخص لرزش را احساس می‌کند. در سایر زمان‌ها بورد آردوینو برای کاهش مصرف انرژی به خواب می‌رود. برای این کار از کتابخانه `LowPower.h` استفاده می‌کنیم. با استفاده از تابع `powerDown` آردوینو به خواب می‌رود. برای بیدارشدن در زمان مشخص،

آردوینو از زمان‌بند داخلی خود<sup>۴</sup> استفاده می‌کند. اما این تایمر بعد از ۸ ثانیه سریز شده و بورد بیدار می‌شود. برای زمان‌های بیشتر باید به محض بیدارشدن مجدد تا رسیدن به زمان مورد نظر، آن را مجدداً به خواب ببریم. این کار در تابع longSleep انجام شده است.

### ۳-۳ دروازه

در دروازه ما از زبان برنامه‌نویسی پایتون برای دریافت فریم‌های زیگبی، ذخیره کردن و ارسال داده‌ها به سرور مرکزی پس از رسیدن به آستانه استفاده می‌کنیم. ساختار پروژه دروازه در شکل ۴-۳ نشان داده شده است. در این بخش به بررسی هر یک از فایل‌ها می‌پردازیم.



شکل ۴-۳: ساختار پروژه دروازه

#### ۱-۳-۳ فایل env\_vars.py

این فایل از بسته load\_dotenv برای خواندن متغیرهای محیطی استفاده می‌کند. برخی از این متغیرها برای برقراری ارتباط با سرور استفاده می‌شوند مانند شماره پورت سرور و آی‌پی سرور. سایر آنها برای برقراری ارتباط با دستگاه ایکس‌بی استفاده می‌شوند. این متغیرهای محیطی به عنوان متغیرهای پایتون تنظیم می‌شوند و سپس در مازول‌های دیگر پایتون استفاده می‌شوند.

#### ۲-۳-۳ فایل models.py

در این فایل ما سه کلاس برای نمایش داده‌های ارتعاش، اندازه گیری‌ها و هر گره انتهایی تعریف کردہ‌ایم.

<sup>7</sup>Watchdog Timer

کلاس داده ارتعاش شامل داده‌های لرزش در سه محور ثبت شده توسط حسگر می‌باشد. هر اندازه‌گیری بسته به فرکانس نمونه‌برداری شامل چندین داده ارتعاش است. همچنین شامل یک شناسه و زمانی است که اندازه‌گیری در آن انجام شده است. از آنجایی که برای برقراری ارتباط داده‌های ارتعاش با زیگبی قطعه‌بندی<sup>۸</sup> لازم است و زمانی بین ارسال هر فریم لازم است، تابع `thread_handler` تعریف شده است تا در صورت اتمام زمان اتصال یا از بین رفتن برخی فریم‌ها، اندازه‌گیری از داده‌های دریافتی قبلی بطور خودکار تکمیل شود. پس از آن داده‌های جدید ذخیره می‌شوند.

هر گره دارای یک شناسه است که با شناسه فرستنده ارائه شده در هر فریم زیگبی مشخص می‌شود. همچنین هر گره شامل چندین اندازه‌گیری است. هنگامی که اندازه‌گیری جدیدی به لیست اندازه‌گیری‌ها اضافه می‌شود، زمان‌بند رشته‌ای<sup>۹</sup> با دوره وقفه مشخص و تابع `thread_handler` شروع می‌شود.

### ۳-۳-۳ فایل `receiver.py`

در این فایل سوروری تعریف می‌کنیم که بطور بی‌نهایت به درگاه USB متصل به دستگاه ایکس‌بی گوش می‌دهد. برای خواندن داده‌ها و استخراج داده هر فریم از کتابخانه `digi-xbee` استفاده می‌کنیم. برای مقداردهی اولیه یک نمونه ایکس‌بی، درگاه و نرخ باود<sup>۱۰</sup> ارتباط لازم است. در تابع `decode_sensor_data` شناسه اندازه‌گیری و داده‌های ارتعاش را استخراج می‌کنیم. در تابع `receive`، به منظور دریافت پیام جدید می‌مانیم. داده‌های پیام را به گره و اندازه‌گیری مربوطه اضافه می‌کنیم و سپس داده‌های ذخیره‌شده را بروز می‌کنیم.

### ۴-۳-۳ فایل `requirements.txt`

این فایل برای تعیین بسته‌ها، نسخه‌ها و وابستگی‌های لازم استفاده می‌شود. سپس از این فایل برای نصب بسته‌های مشخص شده با مدیریت بسته پایتون پیپ<sup>۱۱</sup> استفاده می‌شود.

<sup>8</sup>Fragmentation

<sup>9</sup>Thread Timer

<sup>10</sup>Buad Rate

<sup>11</sup>pip

### ۵-۳-۳ فایل schedule.sh

آستانه تعریف شده بر اساس زمان است. این اسکریپت در کنار گیرنده اجرا می‌شود. در این اسکریپت یک کار crontab برنامه‌ریزی می‌شود که هر ۱۵ دقیقه اجرا شود. این زمان‌بندی با استفاده از علامت  $*/15$  \* \* \* \* انجام می‌شود که به معنای اجرای اسکریپت ارسال داده پایتون در هر دقیقه قابل تقسیم بر ۱۵ است.

### ۶-۳-۳ فایل send\_data.py

در این فایل اطلاعات ذخیره‌شده در حافظه بارگذاری می‌شود. برای سرور و فراخوانی نقاط انتهایی API از کتابخانه requests استفاده می‌کنیم. برای دریافت رمز احراز هویت JWT برای ارسال داده‌های ارتعاشی، یک درخواست پست با آدرس مک دستگاه و رمز عبور به سرور ارسال می‌شود تا رمز احراز هویت را دریافت کند. پس از آن داده‌های لرزش فرمت‌شده بصورت جیسان<sup>۱۲</sup> در بدنه درخواست به همراه هدر احراز هویت به سرور ارسال و فایل ذخیره نیز پاک می‌شود.

### ۷-۳-۳ فایل signup.py

این فایل برای ثبت دروازه در سرور با رمز عبور و آدرس مک مربوطه استفاده می‌شود.

### ۸-۳-۳ فایل utils.py

این فایل شامل برخی از توابع کاربردی مانند دریافت آدرس مک دستگاه، دریافت زمان جاری در شکل مورد نظر، بروز رسانی داده‌های ذخیره‌شده و تعریف کلاس مدل جیسان برای کد کردن داده‌ها می‌باشد.

## ۴-۳ پیش‌پردازش

داده‌های جمع‌آوری شده از حسگر در هر سه محور می‌توانند تحت تاثیر شتاب گرانش زمین باشند. همچنین شتاب اندازه‌گیری شده توسط حسگرهای کم‌هزینه MEMS نیز اکثرا تحت تاثیر یک مقدار غیر صفر دچار انحرافاتی می‌شوند که منجر به افزایش یا کاهش اندازه‌گیری‌ها می‌شود<sup>۱۳</sup>. برای حذف انحرافات و شتاب گرانش ناخواسته، داده‌ها را در مرحله پیش‌پردازش هنجارسازی می‌کنیم. برای حذف

<sup>12</sup>JavaScript Object Notation(JSON)

ناهنجاری‌ها مطابق [برابری \(۱-۳\)](#) میانگین هر محور را از هر داده کم می‌کنیم. لازم به ذکر است که نماد ماتریس هنجارشده برای شتاب در سه محور، گره  $n$  و اندازه‌گیری  $m$  است.

$$\hat{a}_{nm}^l = a_{nm}^l - \sum_{k=1}^K \frac{a_{nmk}^l}{K} \quad (1-3)$$

### ۵-۳ پایگاه داده سری زمانی

مدل داده‌های سری زمانی را در سطلهای <sup>۱۳</sup> و اندازه‌گیری‌ها سازماندهی می‌کند. یک سطل می‌تواند چندین اندازه‌گیری داشته باشد. اندازه‌گیری گروه‌بندی منطقی برای داده‌های سری زمانی است که شامل چندین برچسب <sup>۱۴</sup> و فیلد است. همه نقاط در یک اندازه‌گیری معین باید برچسب‌های یکسانی داشته باشند. برچسب‌ها جفت‌های کلید-مقدار با مقادیری متفاوت هستند که اغلب تغییر نمی‌کنند و برای ذخیره ابرداده <sup>۱۵</sup> برای هر نقطه در نظر گرفته شده‌اند. فیلدها جفت‌های کلید-مقدار هستند که مقادیر آنها در طول زمان تغییر می‌کنند. زمان مرتبط با داده‌ها نیز ذخیره می‌شوند و هنگام پرس‌و‌جو داده‌ها بر حسب آن مرتب می‌شوند <sup>[۱۶]</sup>.

برای ارتباط با پایگاه داده از کلاینت پایتون influxDB استفاده می‌کنیم. برای شروع یک اتصال به نام سطل، سازمان، نشانه و نشانی نیاز داریم. برای نوشتن داده‌ها از تابع `write_api` و برای پرس‌و‌جوی داده‌ها در پایگاه داده از `query_api` استفاده می‌کنیم.

برای نوشتن داده‌ها باید نقاطی را که می‌خواهیم اضافه کنیم، مشخص کنیم. نقطه یک داده واحد است که با اندازه‌گیری، کلیدهای برچسب، مقادیر برچسب، کلید فیلد و زمان مشخص می‌شود. برای پرس‌و‌جوی داده‌ها، از عملگر لوله `glow`<sup>۱۷</sup> برای اتصال خروجی یک تابع بعنوان ورودی تابع بعدی استفاده می‌کند <sup>[۱۸]</sup>. سه تابع اصلی برای پرس‌و‌جوکردن داده‌ها وجود دارد:

- `:from()`: داده‌ها را از یک سطل جستجو می‌کند.

- `:range()`: داده‌ها را بر اساس محدوده‌ی زمانی فیلتر می‌کند. برای جستجو به پرس‌و‌جوهای محدود به یک بازه‌ی زمانی خاص نیاز داریم.

<sup>13</sup>Buckets

<sup>14</sup>Tag

<sup>15</sup>Meta data

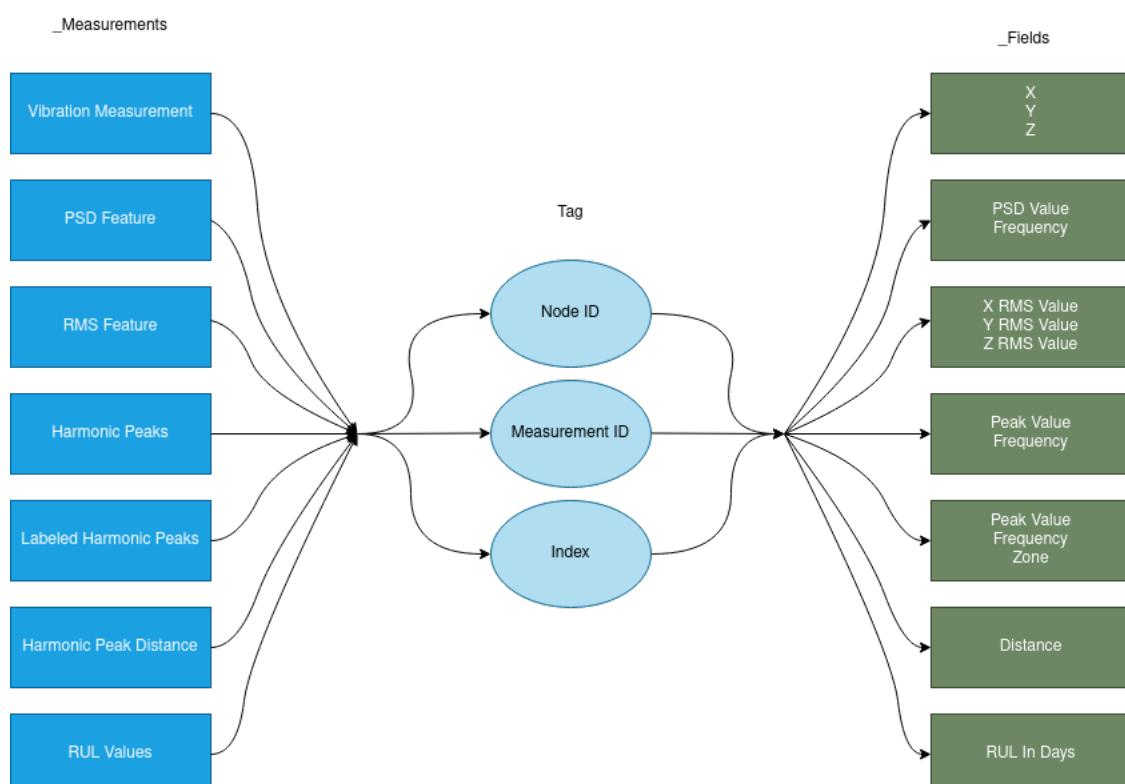
<sup>16</sup>Pipe-Forward(|>)

- داده‌ها را بر اساس مقادیر ستون فیلتر می‌کند. هر سطر با  $r$  و هر ستون با ویژگی  $r$  نشان داده می‌شود. می‌توان چند فیلتر متوالی اعمال کرد.

تابع دیگری که استفاده کردہ‌ایم (`pivot()`) است که برای تبدیل داده‌ها به یک طرح رابطه‌ای<sup>۱۷</sup> بر اساس زمان استفاده می‌شود.

### ۱-۵-۳ ساختار کلی

ساختار کلی پایگاه داده طراحی شده در این پروژه در [شکل ۵-۳](#) نشان داده شده است. هر یک از ویژگی‌های استخراج شده در سمت چپ در اندازه‌گیری‌های مختلف قرار گرفته‌اند. در وسط تصویر نیز برچسب‌ها که بین همه اندازه‌گیری‌ها مشترک است قرار گرفته است. در سمت راست تصویر نیز فیلدهای هر اندازه‌گیری آورده شده است. در ادامه به توضیح توابع نوشتن و پرس‌وجوی اندازه‌گیری‌ها می‌پردازیم.



شکل ۳-۵: ساختار کلی پایگاه داده سری زمانی

<sup>17</sup>Relational Schema

### ۳-۵-۲ اندازه‌گیری‌ها و توابع مربوط به نوشتن و گرفتن داده

در هر اندازه‌گیری، فیلدهای مربوط را ذخیره می‌کنیم. فیلدهای هر داده ثبت می‌شود و با یک گره و یک اندازه‌گیری همراه است. همچنین یک شماره به هر نمونه در اندازه‌گیری عنوان شناسه داده می‌شود. برای نوشتن داده‌ها در پایگاه داده تابع `write_{measurement name}` پیاده‌سازی شده است که یک نقطه جدید برای هر داده با شناسه اندازه‌گیری، شناسه گره و شماره عنوان برچسب و فیلدهای مربوطه ایجاد می‌کند.

برای پرس‌وجوی داده، تابع `get_{measurement name}` پیاده‌سازی شده است. پرس‌وجو بگونه‌ای نوشته شده است که اگر شناسه اندازه‌گیری یا شناسه گره ارائه شود، فقط داده‌های مرتبط برگردانده می‌شود. در هر پرس‌جو، ما تمام داده‌ها را از اندازه‌گیری مربوط انتخاب می‌کنیم و در صورت وجود شناسه گره یا اندازه‌گیری فیلتر می‌کنیم. سپس زمان فیلدها، مقادیر آنها و برچسبها را نگه می‌داریم و از تابع `pivot` برای تبدیل نتایج به طرح رابطه‌ای استفاده می‌کنیم. عنوان نمونه، کد پرس‌وجوی `vibration_measurement` در [شکل ۶-۳](#) نشان داده شده است.

```
filter_by_node: str = f'|> filter(fn:(r) => r.nodeId == "{nodeId}")'
filter_by_measurement: str = f'|> filter(fn:(r) => r.measurementId == "{measurementId}")'

query: str = f'from(bucket:"{INFLUXDB_BUCKET}") \
|> range(start: 0) \
|> filter(fn:(r) => r._measurement == "vibration_measurement") \
{filter_by_node if nodeId is not None else ""} \
{filter_by_measurement if measurementId is not None else ""} \
|> keep(columns: ["_time", "_field", "_value", "nodeId", "measurementId", "index"]) \
|> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value") \
|> yield()'
```

شکل ۳-۶: کد پرس‌وجوی `vibration_measurement`

### ۳-۵-۳ تابع گرفتن شناسه گره و اندازه‌گیری

در این تابع برای گرفتن شناسه‌ها به تنها‌یی، برچسب مورد نظر را نگه می‌داریم و جهت جلوگیری از تکرار با استفاده از تابع `unique()` مقادیر یکتای شناسه را برمی‌گردانیم.

### ۴-۵-۳ تابع `get_starting_service_date`

این تابع برای دریافت زمان اولین داده ثبت شده عنوان مبدا زمانی برای محاسبه زمان سرویس‌شدن تجهیزات نوشته شده است.

### ۳-۵ تابع حذف داده‌ها و حافظه پنهان

برای حذف از کلاس DeleteApi کتابخانه استفاده می‌کنیم. جهت حذف حافظه پنهان، همه اندازه‌گیری‌ها را، بجز آنهایی که خودمان تعریف کرده‌ایم، تا زمان فعلی حذف می‌کنیم. همچنین برای حذف همه داده‌ها با استفاده از تابع clear\_vibration\_data، همه اندازه‌گیری‌ها را حذف می‌کنیم.

### ۳-۶ برنامه وب و چارچوب ویو

وظایف مختلفی در توسعه‌ی یک برنامه وب سمت کاربر وجود دارد. بسته‌های زیادی برای بخشی خاص توسعه داده شده‌اند تا کار توسعه برنامه آسان‌تر شود. در این پژوهه علاوه بر استفاده از چارچوب ویو، از کتابخانه‌ها و مازول‌های دیگر برای کارهای مختلف استفاده کرده‌ایم. در این قسمت در مورد کتابخانه‌های مهم استفاده شده، ساختار فایل پروژه و اجزای اصلی برنامه صحبت و تصاویری از محیط پیاده‌سازی شده را مشاهده می‌کنیم.

### ۳-۶-۱ کتابخانه‌ها و مازول‌ها

#### چارچوب ناکست

ناکست<sup>۱۸</sup> یک چارچوب توسعه وب سطح بالای نودجی اس<sup>۱۹</sup> برای ایجاد برنامه‌های ویو است که می‌تواند در دو حالت مختلف برنامه جهانی<sup>۲۰</sup> و تک صفحه‌ای<sup>۲۱</sup> توسعه و استقرار یابد. علاوه بر این، می‌توان این دو نوع را در ناکست بعنوان برنامه‌های تولیدشده ایستا<sup>۲۲</sup> مستقر کرد. قدرت کامل ناکست در حالت جهانی یا تفسیر سمت سرور نهفته است. ناکست بر روی ویو ایجاد شده است و دارای ویژگی‌های اضافی مانند داده‌های ناهمزمان، میان‌افزار، طرح‌بندی، مازول‌ها و افزونه‌ها است که برنامه را ابتدا در سمت سرور و سپس در سمت کلاینت اجرا می‌کند. در این حالت برنامه معمولاً سریع‌تر از برنامه‌های سنتی سمت سرور ارائه می‌شود<sup>۲۰</sup>. ناکست ساختار فایل ساده‌ای نیز دارد که توسعه برنامه وب را بسیار ساده‌تر می‌کند که در بخش بعدی در مورد آن توضیح می‌دهیم.

<sup>18</sup>Nuxt

<sup>19</sup>Node.js

<sup>20</sup>Universal(SSR)

<sup>21</sup>Single Page Application(SPA)

<sup>22</sup>Static

## کتابخانه ویوتیفای

ویوتیفای<sup>۲۳</sup> یک چارچوب کامل رابط کاربری است که بر روی ویو ساخته شده است. هدف این پروژه ارائه ابزارهای مورد نیاز توسعه‌دهندگان برای ایجاد تجربیات کاربر غنی و جذاب است. برخلاف سایر چارچوب‌ها، ویوتیفای از ابتدا به گونه‌ای طراحی شده است که یادگیری آن آسان است و صدها مؤلفه با دقت ساخته شده از مشخصات طراحی مادی<sup>۲۴</sup> همراه آن است.<sup>[۲۱]</sup>

ویوتیفای برای طراحی از یک رویکرد اول تلفن همراه<sup>۲۵</sup> استفاده می‌کند، به این معنی که برنامه وب به سادگی و بدون بهم ریختگی روی تلفن همراه، تبلت یا رایانه اجرا می‌شود. ویوتیفای را می‌توان به سادگی بعنوان یک ماژول به ناکست اضافه و در پروژه استفاده کرد.<sup>[۲۱]</sup>

## کتابخانه چارت

در این پروژه چندین داده تحلیلی داریم که با استفاده از نمودارها قابل مشاهده هستند. برای این کار از کتابخانه چارت<sup>۲۶</sup> استفاده کرده‌ایم. کتابخانه چارت مجموعه‌ای از انواع نمودارها، افزونه‌ها و گزینه‌های سفارشی‌سازی را ارائه می‌دهد. علاوه بر مجموعه‌ای معقول از انواع نمودار داخل کتابخانه، می‌توان از انواع نمودارهای توسعه‌داده شده توسط جامعه نیز استفاده کرد. علاوه بر این، می‌توان چندین نوع نمودار را در یک نمودار ترکیبی قرار داد. این کتابخانه با افزونه‌های سفارشی ایجاد حاشیه‌نویسی، بزرگنمایی یا قابلیت کشیدن و رها کردن را ممکن ساخته است.<sup>[۲۲]</sup>

## کتابخانه اکسیوس

اکسیوس<sup>۲۷</sup> یک سرویس گیرنده HTTP مبتنی بر وعده<sup>۲۸</sup> برای نودجی‌اس و مرورگر است که می‌تواند در هر دو با پایگاه کد مشابه اجرا شود. این کتابخانه در سمت سرور از ماژول اصلی HTTP نودجی‌اس و در مرورگر از XMLHttpRequests استفاده می‌کند. مزیت اصلی این کتابخانه سادگی در ارسال درخواست‌های مختلف، دریافت پاسخ آنها و پشتیبانی از API وعده است.<sup>[۲۳]</sup> همچنین می‌توان آن را بعنوان یک ماژول به پروژه ناکست اضافه کرد.

<sup>23</sup>Vuetify

<sup>24</sup>Material Design

<sup>25</sup>Mobile First

<sup>26</sup>Chart.js

<sup>27</sup>Axios

<sup>28</sup>Promise

## ماژول احراز هویت ناکست

احراز هویت بخش بسیار مهمی از هر برنامه وب است که شامل مراحل زیر است:

- ارسال اطلاعات کاربران به سرور جهت تایید
- دریافت و ذخیره توکن احراز هویت
- استفاده از آن توکن برای تماس‌های بعدی API

این ماژول با استفاده از یک طرح احراز هویت قابل تنظیم یا با استفاده از یکی از ارائه‌دهندگان مستقیم پشتیبانی شده، احراز هویت کاربران را تأیید می‌کند و یک API برای احراز هویت و دسترسی به اطلاعات کاربر فراهم می‌کند. در حالیکه این ماژول ذخیره اطلاعات در سمت کاربر را بر عهده دارد، مدیریت جلسه یا احراز هویت مبتنی بر جلسه در سمت سرور را ارائه نمی‌کند<sup>[۲۴]</sup>.

## ۲-۶-۳ ساختار پروژه

ساختار پروژه در شکل ۷-۳ نشان داده شده است. برخی از فایل‌ها و پوشه‌ها بخشی از چارچوب ناکست هستند.

برخی دیگر از این فایل‌ها مانند eslint برای هشدار در موقعی که کد نوشته شده مطابق قواعد تعريف شده نباشد استفاده می‌شود، برخی دیگر مانند style lint برای داشتن سبکی یکپارچه و تمیز در بین همه فایل‌ها و توسعه‌دهندگان استفاده می‌شوند.

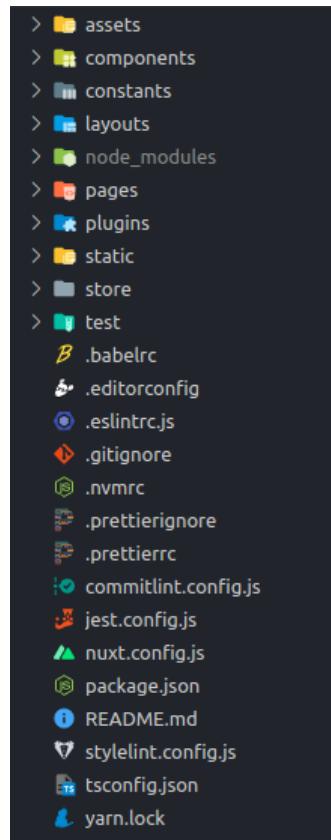
همچنین چند فایل وجود دارد که بسته‌های لازم، نسخه‌های آنها و همچنین چند اسکریپت برای اجرای پروژه را مشخص می‌کنند. مدیریت‌کننده بسته مورد استفاده در این پروژه یارن<sup>۲۹</sup> نام دارد. در فایل yarn.lock نسخه‌های دقیق بسته‌ها و وابستگی‌های آنها توسط یارن مشخص شده است و از آن برای سریع و قابل اعتمادبودن تفکیک وابستگی‌ها استفاده می‌کند. در فایل package.json اسکریپت‌ها، وابستگی‌ها و وابستگی‌های زمان توسعه مشخص شده‌اند. این فایل برای شناسایی بسته‌های لازم توسط یارن در اولین اجرا لازم است.

### پوشه assets

این پوشه شامل فایل‌های تفسیرنشده مثل استایل‌ها<sup>[۳۰]</sup>، تصاویر و فونت‌ها است<sup>[۲۵]</sup>.

<sup>29</sup>Yarn

<sup>30</sup>Styles



شکل ۳-۷: ساختار پروژه برنامه وب

### پوشه components

این پوشه حاوی مؤلفه‌های ویو است. مؤلفه‌ها بخش‌های مختلف صفحه را تشکیل می‌دهند و می‌توانند دوباره استفاده شوند و در صفحات، طرح‌بندی‌ها<sup>۳۱</sup> و حتی سایر مؤلفه‌ها قرار گیرند[۲۵].

### پوشه layouts

برای تغییر ظاهر برنامه، طرح‌بندی‌ها کمک بزرگی هستند. این تغییر می‌تواند افزودن نوار کناری<sup>۳۲</sup> یا طرح‌بندی‌های متفاوت برای تلفن‌همراه و رایانه باشد. در این پروژه دو طرح‌بندی پیش‌فرض و خطا وجود دارد[۲۵].

<sup>31</sup>Layouts

<sup>32</sup>Sidebar

### pages پوشه

این پوشه شامل نماها و مسیرهای برنامه است. ناکست تمام فایل‌های vue. داخل این پوشه را می‌خواند و بطور خودکار پیکربندی مسیریاب را ایجاد می‌کند [۲۵].

### plugins پوشه

این پوشه شامل افزونه‌های جاوا اسکریپت است که می‌خواهیم قبل از نمونه‌سازی ریشه برنامه اجرا کنیم [۲۵]. در این پروژه برای ایجاد مؤلفه نمودارها و همینطور دخالت در درخواست‌های اکسیوس و مدیریت خطاهای مربوط به آن افزونه‌هایی نوشته‌ایم.

### statics پوشه

این پوشه بطور مستقیم به ریشه سرور نگاشت می‌شود و حاوی فایل‌هایی است که تغییر نخواهد کرد. همه فایل‌های ارائه شده بطور خودکار توسط ناکست ارائه<sup>۳۳</sup> می‌شوند و از طریق آدرس ریشه پروژه قابل دسترسی هستند [۲۵].

### store پوشه

گاهی اوقات در پروژه به اطلاعات مشترک و قابل دسترسی بین همه مؤلفه‌ها نیاز داریم. یکی از راهکارهای این مسئله استفاده از سیستم ذخیره‌سازی مرکزی است که در آن یک بخش داده‌های مشخص شده و تغییراتشان را مدیریت می‌کند. در ویو این مسئولیت بر عهده ویوایکس<sup>۳۴</sup> است. این پوشه شامل فایل‌های ویوایکس است. ویوایکس همراه ناکست و بصورت خارج از جعبه<sup>۳۵</sup> عرضه می‌شود اما بطور پیش‌فرض غیرفعال است. ایجاد یک فایل index.js در این پوشه، ویوایکس را فعال می‌کند [۲۵]. در این پروژه برای نگهداری مشخصات ظاهری و مدیریت ورود کاربران از ویوایکس استفاده می‌کنیم.

### nuxt.config.js فایل

بطور پیش‌فرض، ناکست برای پوشش بیشتر موارد استفاده پیکربندی شده است. این پیکربندی پیش‌فرض ناکست و مازول‌های آن را می‌توان با فایل nuxt.config.js بازنویسی کرد [۲۵]. بازنویسی‌های ما شامل

<sup>33</sup>Serve

<sup>34</sup>Vuex

<sup>35</sup>Out of the Box

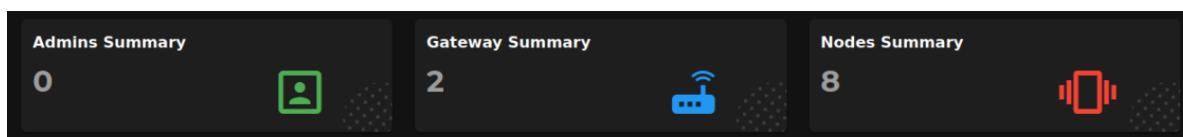
افزودن برچسب‌های ابرداده، مشخص کردن مژول‌ها، افزودنی‌ها و استایل کلی و پیکربندی برخی مژول‌ها مانند مژول احراز هویت(برای مشخص کردن روش احراز هویت، مسیرهای حفاظت شده و آدرس سرور برای تایید احراز هویت) و ویوتیفای(برای مشخص کردن ظاهر صفحات و رنگ‌ها) است.

### ۳-۶-۳ مؤلفه‌های اصلی

در این بخش در مورد اجزای اصلی و برخی از صفحات این پروژه صحبت خواهیم کرد.

#### کارت خلاصه

در این مؤلفه، آمار خلاصه همه موارد مختلف را نشان می‌دهیم. عنوان، خلاصه و نماد بعنوان ورودی<sup>۳۶</sup> ارائه می‌شوند و به سادگی با تکرار روی هر مورد نشان داده می‌شوند. نتایج را می‌توان در [شکل ۳](#) مشاهده کرد که آمار کاربران، گره‌ها و دروازه‌ها را نشان می‌دهد.



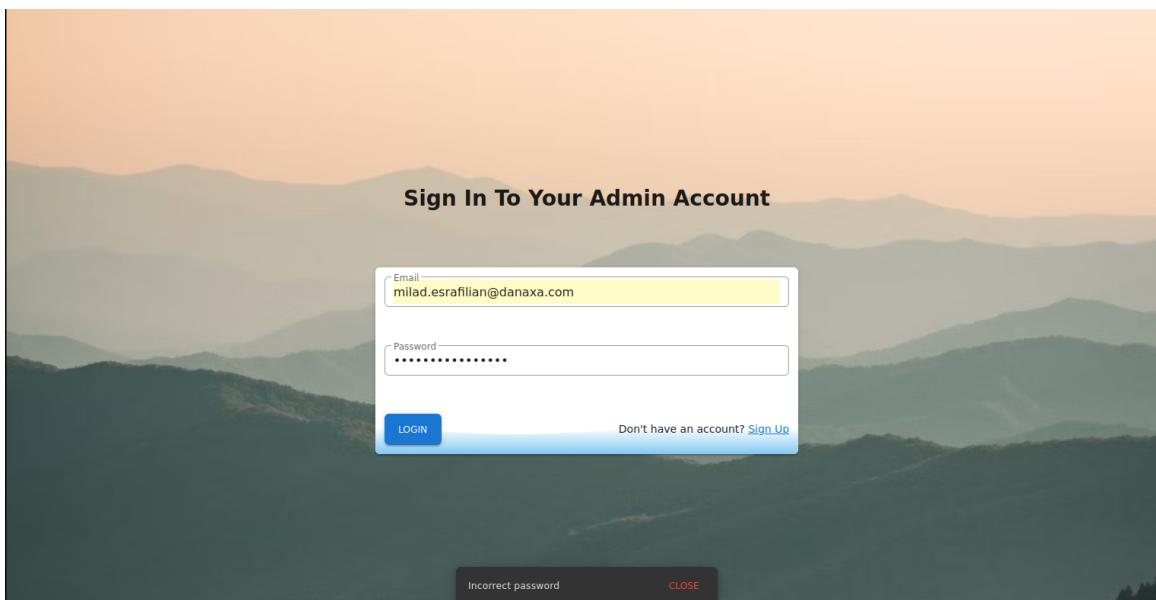
شکل ۳-۳: مؤلفه کارت خلاصه

#### فرم‌ها

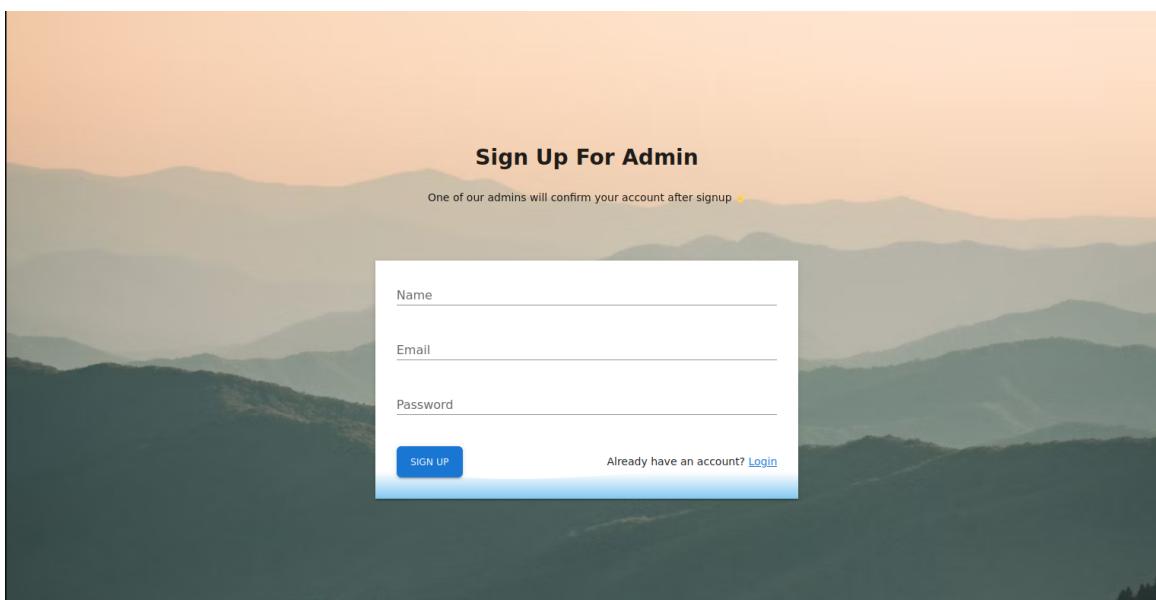
تمامی اجزای مربوط به فرم‌ها در این پوشه قرار دارند. مؤلفه ورود دارای دو فیلد متنی برای ایمیل و رمز عبور کاربران است. هنگامی که کاربر بر روی دکمه ورود کلیک می‌کند، معتبربودن ایمیل و رمز عبور بررسی می‌شود. سپس درخواست مجوز برای کاربر به سرور ارسال می‌شود و در صورت صحیح بودن اطلاعات، کاربر به صفحه اصلی هدایت می‌شود. در غیر این صورت، پیام خطای مناسب با مؤلفه اسنک یار<sup>۳۷</sup> نمایش داده می‌شود. در [شکل ۹-۳](#) صفحه ورود با پیام خطای نادرستی رمز عبور نشان داده شده است. همین روند در مؤلفه ثبت‌نام طی می‌شود، با این تفاوت که یک فیلد نام نیز در آن وجود دارد که در [شکل ۱۰-۳](#) قابل مشاهده است.

<sup>36</sup>Props

<sup>37</sup>SnackBar



شکل ۹-۳: صفحه ورود و پیغام خطای رمز عبور نادرست



شکل ۱۰-۳: صفحه ثبت‌نام کاربر جدید

## اسنکبار

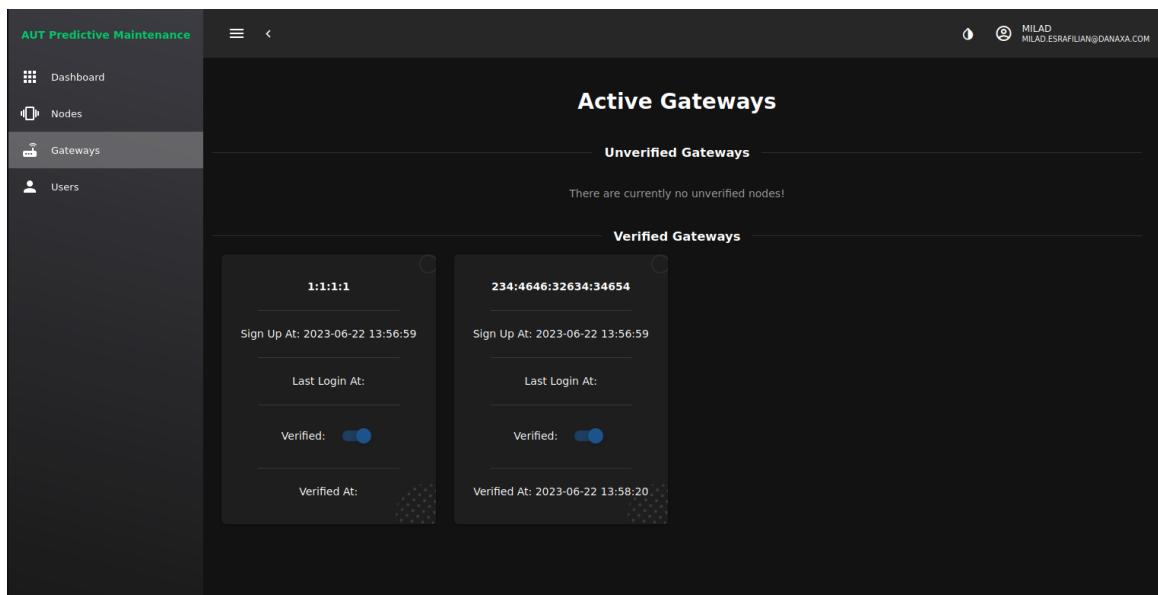
مؤلفه اسنکبار برای نمایش پیام‌های خطا استفاده می‌شود. پیغام خطا به عنوان ورودی به آن داده می‌شود. اسنکبار را می‌توان با فشردن دکمه بستن بست.

## فوتر

فوتر<sup>۳۸</sup> در پایین تمام صفحات نشان داده می‌شود و شامل تاریخ و پیام کپیرایت<sup>۳۹</sup> است.

### مؤلفه‌های UserDetail و GatewayDetail

این دو مؤلفه بسیار شبیه بهم هستند اما اطلاعات متفاوتی را درباره کاربران و دروازه‌ها نشان می‌دهند. هر کاربر و دروازه باید پس از ثبت‌نام توسط مدیر دیگری تایید شود. این کار را می‌توان با تغییر کلید تایید در صفحات مرتبط انجام داد. **شکل ۱۱-۳** دو دروازه فعال تاییدشده را نشان می‌دهد. پس از تایید کاربر یا دروازه، کلید غیرفعال می‌شود.



شکل ۱۱-۳: صفحه دروازه‌های فعال

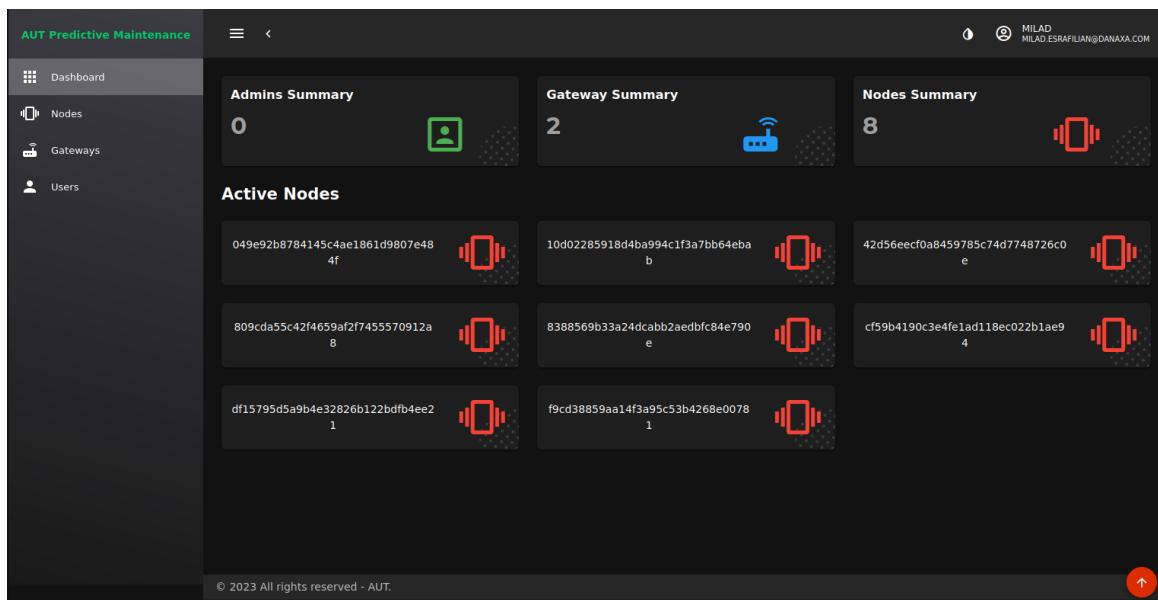
### داشبورد

داشبورد<sup>۴۰</sup> یک نمای کلی از ادمین‌های فعال، دروازه‌ها و گره‌ها ارائه می‌دهد. همچنین در این صفحه می‌توانیم به هر گره مطابق **شکل ۱۲-۳** دسترسی داشته باشیم.

<sup>38</sup>Footer

<sup>39</sup>Copy Right

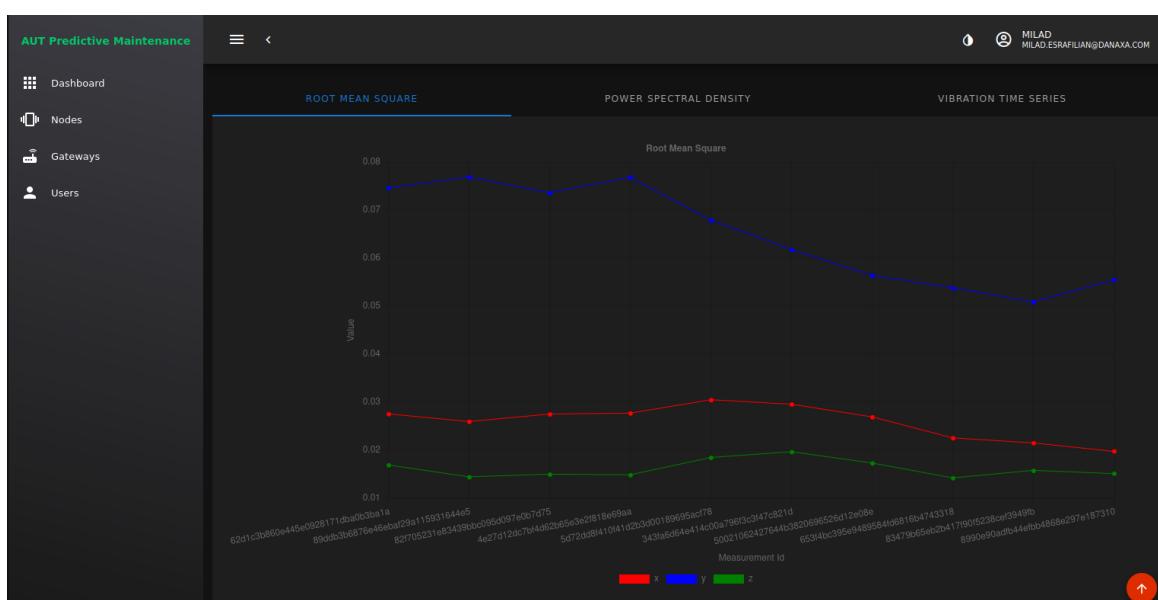
<sup>40</sup>Dashboard



شکل ۱۲-۳: صفحه داشبورد

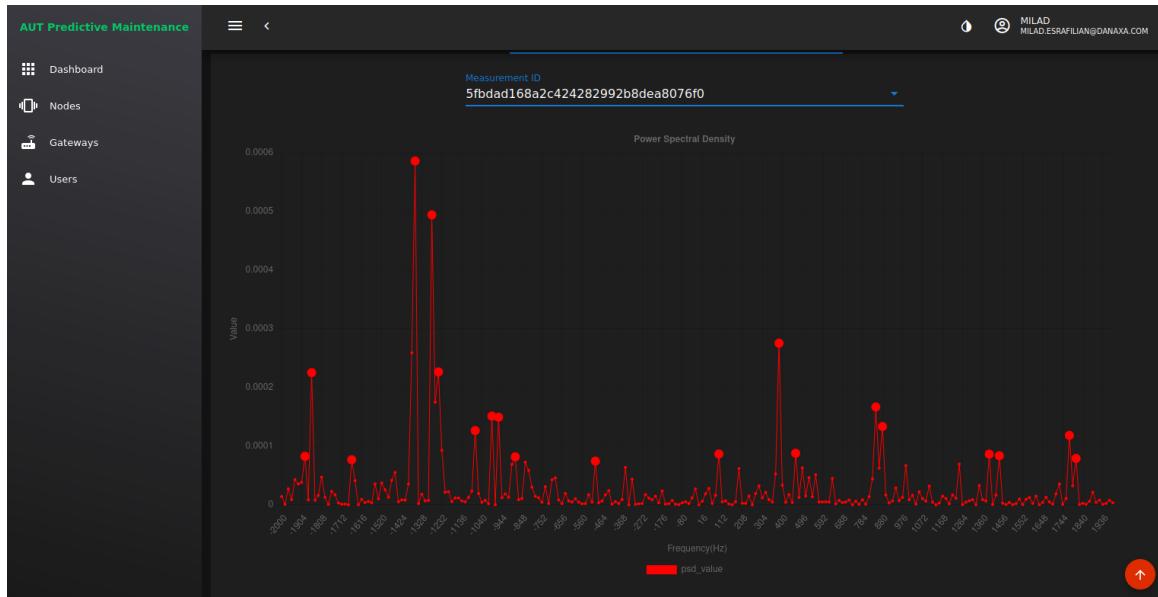
### گره

با کلیک بر روی هر گره، کاربر به صفحه گره مربوط هدایت می‌شود. همانطور که در شکل ۱۳-۳ نشان داده شده است، سه برگه برای مشاهده نمودار تحلیلی مورد نظر وجود دارد. شکل انتخاب پیش‌فرض را نشان می‌دهد که ریشه میانگین مرربع است. هنگامی که این صفحه بارگذاری می‌شود یا زمانی که برگه‌ها تغییر می‌کنند درخواستی به نقطه پایانی مناسب ارسال می‌شود و سپس نمودار مرتبط نشان داده می‌شود. برای نمایش نمودارها از افزونه چارت‌جی‌اس و نمودار خطی استفاده می‌کنیم.



شکل ۱۳-۳: نمودار ریشه میانگین مرربع

برای نمایش نمودار چگالی طیفی توان، ابتدا شناسه اندازه‌گیری باید از ورودی کشویی انتخاب شود. پس از انتخاب، دو درخواست برای دریافت مقادیر PSD و قله‌ها به سرور ارسال می‌شود. سپس قله‌ها مطابق شکل ۱۴-۳ به همراه نمودار چگالی طیفی توان با نقاط بزرگ‌تر نشان داده می‌شوند.



شکل ۱۴-۳: نمودار چگالی طیفی توان

سری زمانی ارتعاش، داده‌های خام سه محور را مطابق شکل ۱۵-۳ در یک نمودار نشان داده می‌شود. در این برگه نیز لازم است ابتدا شناسه اندازه‌گیری انتخاب شود تا نمودار نمایش داده شود.



شکل ۱۵-۳: نمودار سری زمانی ارتعاش

### ۷-۳ نتیجه‌گیری و جمع‌بندی

در این بخش در مورد نحوه پیاده‌سازی سیستم صحبت کردیم. ابتدا در مورد ساختار کلی سیستم صحبت کردیم، سپس در مورد نحوه پیاده سازی کد در آردوینو برای گره‌ها صحبت کردیم. سپس دروازه و تمام فایل‌های آن را توضیح دادیم. سپس در مورد پیش‌پردازش و هنجارسازی داده‌ها صحبت کردیم. پس از آن در مورد طرز کار و کتابخانه پایگاه داده سری زمانی برای نوشتن و چستجوی داده‌ها صحبت کردیم و در پایان در مورد برنامه وب و نحوه کار با آن صحبت کردیم و چند تصویر از خروجی را نیز نشان دادیم.

## فصل چهارم

### چالش‌ها و محدودیت‌ها

در این بخش در مورد چالش‌ها و محدودیت‌هایی که در طول این پروژه با آن مواجه بودیم صحبت خواهیم کرد. برخی از این چالش‌ها مربوط به سخت‌افزار و پروتکل‌ها و برخی دیگر مربوط به نرم‌افزار بودند که در ادامه به آنها می‌پردازیم.

## ۱-۴ چالش‌های سخت‌افزاری

همانطور که در بخش‌های قبلی ذکر شد، بورد آردوینو نانو حافظه بسیار محدودی دارد و همین امر دستیابی به فرکانس‌های نمونه‌برداری بالاتر را غیرممکن می‌کرد، زیرا نمونه‌ها باید در حافظه ذخیره می‌شدند و سپس به دروازه ارسال می‌شد. از آنجایی که ما منابع لازم را نداشتیم، نتوانستیم به فرکانس مورد نظر دست یابیم، اما فرکانس نمونه‌برداری بعنوان یک ثابت در برنامه گره انتها‌یی تعریف شده است و در صورت وجود بورد مناسب قابل تغییر است.

ماژول‌های ایکس‌بی برای ارسال بسته‌های با حجم بیش از ۲۵۶ بایت محدودیت دارند. برای انتقال هر بسته بزرگتر از آن، قطعه قطعه سازی لازم بود. اما ما با آزمایش مشاهده کردیم که باید زمانی بین ارسال هر قطعه در نظر گرفت. در غیر این صورت، برخی از بسته‌ها گاهی گم می‌شوند و حتی با سیاست ارسال مجدد پیش‌فرض ایکس‌بی نیز دریافت نمی‌شوند. این مسئله نیازی جدید برای تعریف بازه زمانی مناسب در سمت گیرنده برای فعال‌ماندن و تشخیص قطعات مختلف ایجاد کرد.

## ۲-۴ چالش‌های نرم‌افزاری

همانطور که قبلاً ذکر کردیم، کتابخانه‌های موجود برای استفاده از ماژول‌های ایکس‌بی مشکلات فراوانی داشتند. نوشتمن یک کتابخانه جدید حتی برای استفاده محدود در این زمینه چالشی واقعی بود زیرا باید در سطوح پایین و با عملیات‌های بیتی پیاده‌سازی می‌شد.

اشکال‌زدایی<sup>۱</sup> یکی دیگر از مسائل مهم بود. ماژول ایکس‌بی به درگاه سریال آردوینو متصل است و درگاه USB روی آردوینو که برای تغذیه یا برنامه‌ریزی استفاده می‌شود نیز به همان درگاه متصل است. بنابراین استفاده همزمان از مانیتور سریال<sup>۲</sup> و ماژول ایکس‌بی ممکن نبود و برای برنامه‌ریزی آردوینو، مجبور بودیم ماژول را از جدا کنیم.

<sup>1</sup> Debugging

<sup>2</sup> Serial Monitor

همچنین ماژول ایکس‌بی ما بسیار قدیمی بود. برای برنامه‌ریزی خود ماژول، باید از سیستم‌افزار<sup>۳</sup> ساخته شده سازنده استفاده می‌کردیم و از آنجایی که ماژول بسیار قدیمی بود، یافتن و دریافت سیستم‌افزاری که با ماژول ما سازگار باشد دشوار بود.

### ۳-۴ جمع‌بندی و نتیجه‌گیری

در این بخش درباره چالش‌ها و محدودیت‌های مختلف سخت‌افزاری و نرم‌افزاری که هنگام پیاده‌سازی پروژه با آن مواجه شدیم صحبت کردیم.

---

<sup>3</sup>Firmware

## فصل پنجم

### جمع‌بندی و نتیجه‌گیری و پیشنهادها

## ۱-۵ جمع‌بندی و نتیجه‌گیری

در این پژوهه سعی شد بخشی از یک سیستم پیش‌بینی طول عمر تجهیزات صنعتی طبق [۵] و بر اساس داده‌های ارتعاشی پیاده‌سازی شود.

در فصل اول سعی شد مقدمه‌ای برای نگهداری و معرفی روش‌های مختلف نگهداری تجهیزات در محل‌های صنعتی امروزی بیان شود.

در فصل دوم اصول و تجهیزات مورد استفاده را توضیح دادیم و در مورد حسگر ارتعاش MEMS، آردوبینو نانو بعنوان گره پایانی، پروتکل زیگبی و ماژول ایکس‌بی، رزبری‌پای و استفاده از آن بعنوان دروازه InfluxDB بعنوان پایگاه داده سری زمانی، کتابخانه نامپای و پیش‌پردازش و چارچوب ویو برای توسعه برنامه وب صحبت کردیم.

در فصل سوم نحوه پیاده‌سازی سیستم را توضیح داده و یک نمای کلی از نحوه عملکرد سیستم ارائه دادیم. سپس کتابخانه‌ها، توابع، فایل‌ها و هدف آنها را در این پژوهه توضیح دادیم. در پایان نیز چند تصویر از نتیجه برنامه وب نشان دادیم.

در فصل چهارم در مورد چالش‌ها و محدودیت‌هایی که در اجرای پژوهه در بخش نرم‌افزار و سخت‌افزار با آن مواجه بودیم، صحبت کردیم.

## ۲-۵ پیشنهادها

این پژوهه می‌تواند برای پیش‌بینی دقیق‌تر و بهینه‌تر بهبود یابد. در این بخش پیشنهادهای خود برای بهبود پژوهه را توضیح می‌دهیم.

## ۱-۶ حل محدودیت‌های سخت‌افزاری

ما می‌توانیم از سایر بوردهای آردوبینو که حافظه بیشتری دارند برای دستیابی به فرکانس‌های نمونه‌برداری بالاتر استفاده کنیم، اما باید مراقب مصرف انرژی و هزینه نیز باشیم، زیرا این بوردها با باتری و در تعداد زیاد بعنوان گره‌های انتهایی کار خواهند کرد.

## ۲-۲-۵ پیاده‌سازی ارسال اعلان و برنامه تلفن همراه

برای بهبود تجربه کاربری و اطلاع کاربران از هرگونه خرابی در تجهیزات، می‌توان با پیاده‌سازی نسخه تلفن همراه برنامه وب و ویژگی ارسال اعلان هشدار به کاربران را آسان‌تر کرد.

## ۳-۲-۵ تکمیل کتابخانه ایکس‌بی

نسخه ما از کتابخانه فقط برای یک نوع بسته پیاده‌سازی شده است. با پیاده‌سازی انواع دیگر فریم‌ها، اجرای سیاست کنترل خطای دیگری علاوه بر زیگبی ممکن می‌شود. به این ترتیب می‌توانیم کنترل و بازیابی خطای لایه برنامه را نیز پیاده‌سازی کنیم.

## ۴-۲-۵ استفاده از سایر معیارهای محیطی

اگرچه دما و رطوبت معیارهای اندازه‌گیری دقیقی برای کاربرد ما نیستند و به محیط اطراف خود وابسته‌اند، استفاده از آنها همراه با لرزش می‌تواند دقیق پیش‌بینی‌ها را بهبود دهد.

## منابع و مراجع

- [1] Zhao, Jingyi, Gao, Chunhai, and Tang, Tao. A review of sustainable maintenance strategies for single component and multicomponent equipment. *Sustainability*, 14(5):2992, 2022.
- [2] Tinga, Tiedo. Application of physical failure models to enable usage and load based maintenance. *Reliability engineering & system safety*, 95(10):1061–1075, 2010.
- [3] Wu, Sze-jung, Gebraeel, Nagi, Lawley, Mark A, and Yih, Yuehwern. A neural network integrated decision support system for condition-based optimal predictive maintenance policy. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(2):226–236, 2007.
- [4] Kaiser, Kevin A and Gebraeel, Nagi Z. Predictive maintenance management using sensor-based degradation models. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(4):840–849, 2009.
- [5] Jung, Deokwoo, Zhang, Zhenjie, and Winslett, Marianne. Vibration analysis for iot enabled predictive maintenance. in *2017 ieee 33rd international conference on data engineering (icde)*, pp. 1271–1282. IEEE, 2017.
- [6] Carlsson, Anders, Kuzminykh, Ievgeniia, Franksson, Robin, and Liljegren, Alexander. Measuring a lora network: Performance, possibilities and limitations. in *Internet of Things, Smart Spaces, and Next Generation Networks and*

- Systems: 18th International Conference, NEW2AN 2018, and 11th Conference, ruSMART 2018, St. Petersburg, Russia, August 27–29, 2018, Proceedings 18*, pp. 116–128. Springer, 2018.
- [7] Tabbane, Sami. IoT systems overview. *International Telecommunication Union*, 2017.
- [8] Ramya, C Muthu, Shanmugaraj, Madasamy, and Prabakaran, R. Study on zigbee technology. in *2011 3rd international conference on electronics computer technology*, vol. 6, pp. 297–301. IEEE, 2011.
- [9] Gislason, Drew. *Zigbee wireless networking*. Newnes, 2008.
- [10] Song, SM and Yao, WJ. Research on the application value of wireless mesh network in power equipment of the upiot. in *Journal of Physics: Conference Series*, vol. 1346, p. 012046. IOP Publishing, 2019.
- [11] Salih, Mohammed A Abdala& Alaa Mohammed. Design and performance analysis of building monitoring system with wireless sensor networks. *Iraqi Journal of Science*, 53(4):1097–1102, 2012.
- [12] International, Digi. Xbee zigbee guide. <https://www.digi.com/resources/documentation/Digidocs/90001942-13/Default.htm>, 2022.
- [13] Raspberry pi datasheet. <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf>.
- [14] Naqvi, Syeda Noor Zehra, Yfantidou, Sofia, and Zimányi, Esteban. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles*, 12, 2017.
- [15] Influxdb. <https://www.influxdata.com/time-series-platform/>, Jun 2023.
- [16] Bressert, Eli. Scipy and numpy: an overview for developers. 2012.

- [17] Harris, Charles R, Millman, K Jarrod, Van Der Walt, Stéfan J, Gommers, Ralf, Virtanen, Pauli, Cournapeau, David, Wieser, Eric, Taylor, Julian, Berg, Sebastian, Smith, Nathaniel J, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- [18] Vue.js. <https://vuejs.org/guide/introduction.html>.
- [19] Influxdb document. <https://docs.influxdata.com/influxdb/v2.7/>, Jun 2023.
- [20] Kok, Lau Tiam. *Hands-on Nuxt.js Web Development: Build universal and static-generated Vue.js applications using Nuxt.js*. Packt Publishing Ltd, 2020.
- [21] Vuetify document. <https://v2.vuetifyjs.com/en/introduction/why-vuetify/>, Aug 2023.
- [22] Chart.js document. <https://www.chartjs.org/docs/latest/>, Apr 2023.
- [23] Axios-http document. <https://axios-http.com/docs/intro>.
- [24] Nuxt/auth document. <https://auth.nuxtjs.org/>, Dec 2021.
- [25] Nuxt.js document. <https://v2.nuxt.com/docs>, Jun 2023.

## پیوست

همه کدها و مستندات بخش گره نهایی، کتابخانه ایکس‌بی و دروازه در مخزن زیر در گیتهاب قابل دسترسی است:

<https://github.com/mies47/Predictive-Maintenance-Gateway>

بخش پیش‌پردازش و همینطور راهاندازی و توابع نوشتن و جستجوی پایگاه داده سری زمانی، در مخزن زیر که مربوط به سرور است قابل دسترسی است:

<https://github.com/mies47/Predictive-Maintenance-Server>

بخش برنامه وب به همراه مستندات نصب و اجرای آن نیز در مخزن مجازی زیر قابل مشاهده است:

<https://github.com/mies47/Predictive-Maintenance-Web-App>

# **Abstract**

In recent years, the internet of things has become one of the most popular technology topics. The application of this technology in all areas of human life, as well as recent developments in the fields of data collection, network and artificial intelligence, have made the internet of things to become the focus of many researchers. One of the challenges in industries and factories is the optimal replacement of parts. Due to the lack of sufficient information to analyze the condition of the parts, the appropriate solution to ensure the operation of the production line is to use experts to inspect the condition of the equipment or replace them regardless of their current conditions and only according to a previous schedule. In addition to not being accurate, these solutions impose many expenses and time on industries. In this project, we intend to provide a framework for analyzing equipment vibration data by using a combination of internet of things and artificial intelligence, so that it can be used to predict the remaining useful lifetime of parts. Also, using the web-based dashboard, the results can be shown to experts so that they can provide detailed planning for equipment maintenance and repair. By using this approach, the cost and time spent on replacing parts will be significantly reduced.

## **Key Words:**

Internet of Things, Predictive Maintenance, Industrial Internet of Things, Cyber-Physical Systems Maintenance



**Amirkabir University of Technology  
(Tehran Polytechnic)**

**Department of Computer Engineering**

**B. Sc. Thesis**

# **A Vibration Sensor Data Gathering System for IoT-based Predictive Maintenance**

**Author**

**Arian Boukani**

**Supervisor**

**Dr. Hamidreza Zarandi**

**July 2023**