



دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی

پیاده‌سازی درواره‌ی ارتباطی مرکزی برای سیستم اندازه‌گیری  
هوشمند مصرف در ساختمان‌ها

نگارش

بهار کاویانی

استاد راهنما

دکتر مرتضی صاحب‌الزمانی

۱۴۰۲ مهر

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



دانشگاه صنعتی امیرکبیر  
(پلی‌تکنیک تهران)

به نام خدا

## تعهدنامه اصالت اثر

تاریخ: مهر ۱۴۰۲

اینجانب بهار کاویانی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظرارت و راهنمایی استادی 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، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

بهار کاویانی

امضا

تقدیم به مادر عزیزو و مهربانم، که همواره محکم ترین پناهگاه زندگی ام بودی و باگرمایی  
که از عشقت به دلم رسید، شور زندگی و توان ادامه دادن را به من بخشیدی.

# سپاسگزاری

از استاد دلسوز و محترم؛ جناب دکتر مرتضی صاحب الزمانی و همچنین استادیار ایشان، جناب آقای ملکوتی که با صبر و حوصله، از هیچ کمکی در مسیر انجام این پروژه و نوشتن این پایان‌نامه از من دریغ ننمودند؛ کمال تشکر و قدرانی را دارم.

بهار کاویانی  
مهر ۱۴۰۲

## چکیده

یکی از فعالیت‌های کلیدی برای افزایش کیفیت تأمین مصارف مختلف خانگی همچون آب، برق و گاز، دقت در اندازه‌گیری مصرف آن‌ها است. هر ساله زمان و مبالغ بسیار زیادی به دلیل غیرهوشمند بودن این موضوع، صرف جمع‌آوری و خواندن داده‌های کنتورها در سراسر کشور می‌شود. توسعه‌ی حسگرها، دستگاه‌ها و هوشمندسازی روندهای مکانیکی و تکراری برای اندازه‌گیری مصارف خانگی، پایه‌ای برای ارائه با کیفیت‌تر این خدمات است.

با جایگزین کردن کنتورهای هوشمند به جای کنتورهای عادی، مشکلات بسیاری را می‌توان بهبود بخشد؛ از جمله هزینه‌ی جمع‌آوری اطلاعات تمام کنتورهای یک شهر و استان، هزینه‌های ناشی از خطای انسانی، صدور قبض‌های ماهانه و نداشتن اطلاعات از میزان مصرف شهروندان به دلیل نبود سیستمی در راستای جمع‌آوری لحظه‌ای این اطلاعات.

برای طراحی چنین سیستمی که بتواند جایگزین روش امروزی در خواندن داده‌های کنتورها در ساختمان‌ها باشد، علاوه بر سخت‌افزار متصل شده به کنتور و همچنین نرم‌افزاری که اطلاعات جمع‌آوری شده را در اختیار کاربران قرار دهد، وجود یک درگاه مرکزی برای جمع‌آوری اطلاعات و ارسال آن‌ها الزامی خواهد بود. این بخش مرکزی باید قادر باشد که اطلاعات جمع‌آوری شده را به گونه‌ای پردازش کند که قابل استفاده برای مصارف اینترنت اشیا باشد. تمرکز این پروژه بر روی دریافت اطلاعات خوانده‌شده توسط کنتور و ارسال آن‌ها به سمت سرورهایی جهت ذخیره‌ی این اطلاعات است. این موضوع می‌تواند بهبود مهمی در مدیریت مصرف انرژی و خدمات ارائه‌شده توسط سازمان‌های مرتبط را به دنبال داشته باشد.

## واژه‌های کلیدی:

اینترنت اشیا، سامانه‌ی اندازه‌گیری هوشمند، کنتورهای هوشمند، اندازه‌گیری مصارف خانگی، بهینه‌سازی

# فهرست مطالب

صفحه

عنوان

۱	مقدمه . . . . .	۱
۲	۱-۱ ظهور اینترنت اشیا . . . . .	
۳	۲-۱ کنتورهای هوشمند . . . . .	
۴	۳-۱ دربارهی پروژه . . . . .	
۶	۲ مفاهیم پایه . . . . .	۲
۷	۱-۲ بردهای مورد استفاده در پروژه . . . . .	
۷	۱-۱-۱ برد M5Stack با پردازندهی ESP32 . . . . .	۱-۲
۸	۱-۱-۲ مقایسه و انتخاب برد اصلی پروژه . . . . .	
۱۰	۲-۱-۱-۲ ویژگی‌ها و نکات مرتبط با برد منتخب . . . . .	
۱۱	۲-۱-۲ برد ESP8266 . . . . .	
۱۳	۳-۱-۲ مازول فرستنده و گیرنده RS485 . . . . .	
۱۴	۲-۲ پروتکل‌ها و مفاهیم برای ارتباط با بخش نرمافزار . . . . .	۲
۱۴	۱-۲-۲ پروتکل MQTT . . . . .	
۱۴	۱-۱-۲-۲ کارگزار MQTT . . . . .	
۱۶	۲-۱-۲-۲ مقایسه کارگزارهای MQTT . . . . .	
۱۷	۲-۲-۲ زیرساخت ارتباطی با بخش نرمافزار . . . . .	
۱۷	۱-۲-۲-۲ زیرساخت ماشین مجازی . . . . .	
۱۸	۲-۲-۲-۲ سرویس داکر . . . . .	
۱۸	۳-۲-۲-۲ چارچوب نرمافزاری جنگو . . . . .	
۱۹	۳-۲ پروتکل‌ها و مفاهیم برای ارتباط با بخش سختافزار . . . . .	
۱۹	۱-۳-۲ استاندارد RS485 . . . . .	
۲۱	۲-۳-۲ پروتکل Modbus . . . . .	
۲۲	۳ طراحی و پیاده‌سازی دروازهی مرکزی . . . . .	
۲۳	۱-۳ اتصال برد اصلی به کارگزار MQTT . . . . .	
۲۷	۲-۳ اتصال برد شبیه‌ساز به برد اصلی . . . . .	
۳۲	۳-۳ ارسال داده‌ها از کارگزار MQTT به پایگاه داده و ذخیره‌ی اطلاعات . . . . .	
۳۲	۱-۳-۳ طراحی و راهاندازی پایگاه داده MySQL . . . . .	
۳۴	۲-۳-۳ طراحی و راهاندازی برنامه‌ی پسین بر روی چارچوب نرمافزاری جنگو . . . . .	
۳۶	۴ چالش‌ها و محدودیت‌های انجام پروژه . . . . .	
۳۷	۱-۴ چالش‌های فرایندی . . . . .	

۳۷	۱-۱-۴ تعیین فرایند برای چگونگی ارتباط با سامانه‌ی نرم‌افزاری	۴
۳۷	۲-۱-۴ انتخاب پروتکل برای ارتباط با کنترل سخت‌افزاری	۴
۳۸	۲-۴ چالش‌های معماری	۴
۳۸	۱-۲-۴ انتخاب موضوعات مناسب بر روی پروتکل MQTT	۴
۳۹	۲-۲-۴ امنیت داده‌های ارسالی	۴
۴۰	۳-۴ چالش‌های سخت‌افزاری	۴
۴۰	۱-۳-۴ تولید ورودی‌های آزمون	۴
۴۱	۲-۳-۴ برقراری اتصالات مناسب	۴
۴۱	۳-۳-۴ پیکربندی ارتباط سریال	۴
۴۲	۴-۳-۴ نحوه‌ی بررسی ارسال داده بین دو برد	۴
۴۳	۴-۴ چالش‌های نرم‌افزاری	۴
۴۳	۱-۴-۴ نحوه‌ی بررسی ارسال داده بین برد اصلی و کارگزار MQTT	۴
۴۳	۲-۴-۴ لزوم آشنایی کامل با سخت‌افزار برد در هنگام توسعه‌ی برنامه	۴
۴۴	۵ جمع‌بندی و پیشنهادها	۵
۴۵	۱-۵ جمع‌بندی	۵
۴۶	۲-۵ پیشنهادها و کارهای آینده	۵
۴۸	مراجع	۵
۵۱	واژه‌نامه‌ی فارسی به انگلیسی	۵
۵۴	واژه‌نامه‌ی انگلیسی به فارسی	۵

صفحه	فهرست تصاویر	شکل
		۱
۴	سیستم اندازه‌گیری هوشمند . . . . .	
۵	نمودار ارتباطات و بخش‌های مختلف پروژه . . . . .	
۸	نمونه‌ای از یک برد آردوینو . . . . .	۱-۲
۱۱	برد M5Stack مدل ساده . . . . .	۲-۲
۱۱	طرح‌واره‌ی درگاه‌های متصل به هسته‌ی برد M5Stack . . . . .	۳-۲
۱۲	طرح‌واره‌ی درگاه‌های متصل به برد ESP8266 . . . . .	۴-۲
۱۳	تصویر ماژول فرستنده و گیرنده RS485 . . . . .	۵-۲
۱۵	نحوه ارتباط کارخواه‌ها با یکدیگر به کمک کارگزار MQTT [۱] . . . . .	۶-۲
۲۰	شبکه‌ی نیمه دوطرفه RS485 . . . . .	۷-۲
۲۰	شکل موج RS485 . . . . .	۸-۲
۲۴	قطعه کد مربوط به راهاندازی برد M5Stack . . . . .	۱-۳
۲۴	قطعه کد مربوط به اتصال برد به وای-فای . . . . .	۲-۳
۲۵	قطعه کد رویدادهای اتصال به وای-فای . . . . .	۳-۳
۲۶	قطعه کد اتصال به کارگزار MQTT . . . . .	۴-۳
۲۹	تصویر کامل اتصالات . . . . .	۵-۳
۳۰	تصویر اتصالات برد ESP8266 به ماژول فرستنده و گیرنده RS485 . . . . .	۶-۳
۳۱	تصویر اتصالات دو ماژول RS485 . . . . .	۷-۳
۳۱	تصویر اتصالات برد M5Stack به ماژول فرستنده و گیرنده RS485 . . . . .	۸-۳
۳۳	جدول پایگاه داده . . . . .	۹-۳
۳۳	محتوای فایل داکر کامپوز پایگاه داده . . . . .	۱۰-۳
۳۴	مدل طراحی شده در اپلیکیشن records . . . . .	۱۱-۳
۳۵	تنظیمات اتصال به کارگزار MQTT در چارچوب جنگو . . . . .	۱۲-۳
۴۰	تصویری از محیط نرم‌افزار MQTT Explorer . . . . .	۱-۴

## فهرست جداول

صفحه

جدول

- |     |  |    |
|-----|--|----|
| ۱-۳ | جدول پایه‌های ماژول فرستنده و گیرنده RS485               | ۲۸ |
| ۲-۳ | جدول اتصالات برد ESP8266 به ماژول فرستنده و گیرنده RS485 | ۳۰ |
| ۳-۳ | جدول اتصالات برد M5Stack به ماژول فرستنده و گیرنده RS485 | ۳۱ |

# فصل اول

## مقدمه

در این فصل، ابتدا به ارائه‌ی مقدماتی درباره‌ی اینترنت اشیا<sup>۱</sup> و خانه‌های هوشمند<sup>۲</sup> و سپس به لزوم وجود کنتورهای هوشمند خواهیم پرداخت. سپس در انتها هدف و محدوده‌ی پروژه را شرح خواهیم داد.

## ۱-۱ ظهور اینترنت اشیا

امروزه اهمیت فناوری اینترنت اشیا و گسترش قابل توجه آن در سال‌های اخیر بر کسی پوشیده نیست. اینترنت اشیا به زبان ساده مجموعه‌ای از اشیا و تجهیزات است که در یک شبکه با یکدیگر در ارتباط هستند و هر کدام از آن‌ها مجهز به حسگرها<sup>۳</sup> و عملگرهایی<sup>۴</sup> هستند که می‌توانند شرایط مختلف محیط را اندازه‌گیری و پردازش کنند<sup>[۲]</sup>. این فناوری با تخصیص شناسه‌ی منحصر به فرد به هر یک از اشیای موجود در یک شبکه، برقراری ارتباط این اشیا را بر بستر اینترنت تسهیل می‌کند. این موضوع سبب می‌شود تا انتقال اطلاعات بین اشیا بدون دخالت انسان صورت گیرد و در نتیجه سرعت و دقیقت در مدیریت داده‌ها را برای ما به ارمغان می‌آورد<sup>[۳]</sup>.

یکی از حوزه‌های مهم و کاربردی اینترنت اشیا، شبکه‌ی هوشمند<sup>۵</sup> و خانه‌ها و شهرهای هوشمند هستند. این موضوع، به ویژه در شهرهای بزرگ و پرجمعیت به دلیل چالش‌های ناشی از روابط متنوع و بالا بودن پیچیدگی سیستم، بیشتر خود را نشان می‌دهد. برای مدیریت و کاهش هزینه‌ی این چالش‌ها، دیگر روش‌های سنتی در چنین شهرهایی پاسخ‌گو نیستند و هوشمندسازی به یکی از نیازهای اصلی تبدیل می‌شود.

در یک ساختمان هوشمند نیز به همین ترتیب ارتباط‌های پیچیده‌ای ممکن است وجود داشته باشد که به تبع آن به مجموعه‌ای از دستگاه‌های هوشمند نیاز خواهد بود تا در خودکارسازی روندهای انسانی به ما کمک کنند. از جمله دستگاه‌های موجود در یک خانه‌ی هوشمند می‌توان به اندازه‌گیری میزان مصارف به طور خودکار و هوشمند اشاره کرد. در ادامه به توضیح کامل این بخش خواهیم پرداخت. این اندازه‌گیری‌ها نه تنها به مدیریت دقیق مصرف‌ها کمک می‌کنند بلکه در کاهش مصرف نیز نقش اساسی دارند. با اطلاعات دقیق از میزان مصرف‌ها، می‌توان بهینه‌سازی استفاده از منابع را انجام داد و در نتیجه به کاهش هزینه‌ها و حتی حفاظت از محیط زیست کمک کرد. همچنین، این اطلاعات می‌توانند

<sup>1</sup>Internet of Things (IoT)

<sup>2</sup>Smart Home

<sup>3</sup>Sensors

<sup>4</sup>Actuators

<sup>5</sup>Smart Grid

به مسئولان و تصمیم‌گیرندگان کمک کنند تا سیاست‌های مدیریت مصرف منابع را با دانش بیشتری پیاده‌سازی کنند و برنامه‌هایی برای حفاظت از زیست‌بوم اطراف خلق کنند. بنابراین، فناوری اینترنت اشیا نه تنها می‌تواند ما را در بهره‌وری انرژی یاری کند بلکه در مدیریت مصرف منابع و حفاظت از محیط زیست نیز نقش کلانی ایفا می‌کند. این تعامل میان فناوری و محیط زیست، گام مهمی در جهت تحقق شهرها و خانه‌های هوشمند و پایدار است.

## ۲-۱ کنتورهای هوشمند

کنتور هوشمند یا سیستم اندازه‌گیری هوشمند<sup>۱</sup>، یک دستگاه الکترونیکی است که اطلاعات مربوط به مصرف انرژی را اندازه‌گیری می‌کند. کنتورهای هوشمند این اطلاعات را به منظور شفافیت در نوع و میزان مصرف با مشتریان به اشتراک می‌گذارند<sup>[۴]</sup>.

امروزه با رشد جمعیت، مصرف انرژی و منابعی همچون آب، برق و گاز رو به افزایش است. آگاهی‌بخشی به مصرف کنندگان از میزان مصرف خود و امکان بررسی مداوم، باعث می‌شود تا افراد به منظور کاهش هزینه‌های ناشی از مصرف، از این منابع بهتر استفاده کنند. به علاوه، سازمان‌های ارائه‌دهنده‌ی این انرژی‌ها و منابع، با دانستن میزان مصرف مشتریان خود، می‌توانند به داده‌های مفیدی از جمله زمان اوج مصرف، عوامل تاثیرگذار در مصرف مشتریان و دیگر داده‌های مرتبط، دست یابند<sup>[۵]</sup>.

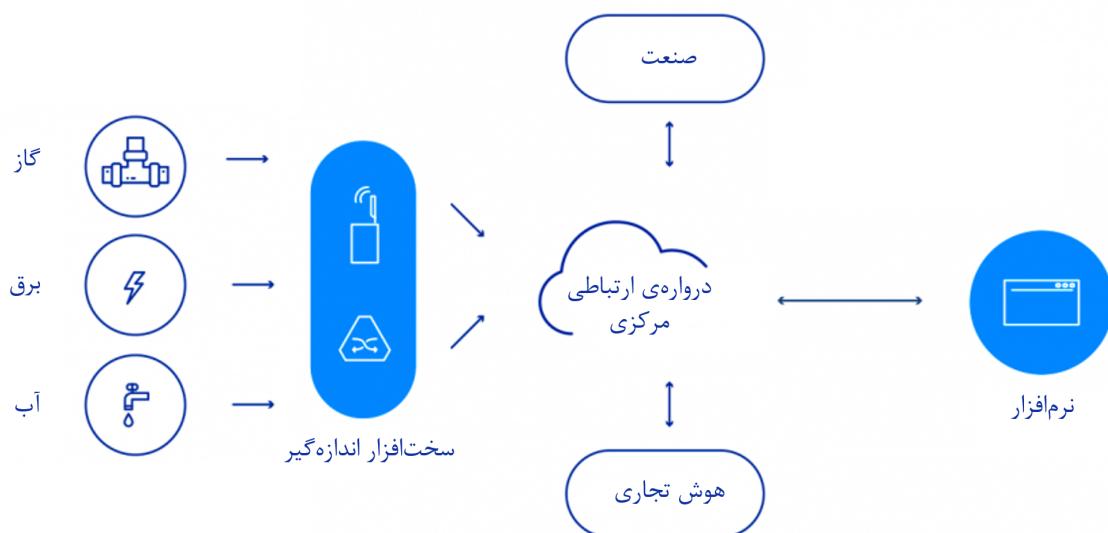
هر کنتور هوشمند نیازمند سه بخش اساسی در ساختار خود است تا بتواند در شبکه‌ی اینترنت اشیا، فرایند خودکارسازی را از ابتدا تا انتهای به درستی پیش ببرد. این سه بخش شامل موارد زیر است:

- سخت‌افزار اندازه‌گیر: این بخش وظیفه دارد تا میزان مصرف انرژی و منابع را به درستی اندازه‌گیری کرده و آن اطلاعات را در اختیار بخش درواوه‌ی ارتباطی مرکزی قرار دهد.
- درواوه‌ی ارتباطی مرکزی<sup>۲</sup>: این بخش باید قادر باشد که اطلاعات جمع‌آوری شده توسط کنتور را به گونه‌ای پردازش کند که قابل استفاده برای مصارف اینترنت اشیا باشد و به راحتی در اختیار مصرف کنندگان یا سازمان‌های ارائه‌دهنده‌ی خدمات مربوط به انرژی و منابع قرار گیرد.
- نرم‌افزار: این بخش وظیفه دارد تا اطلاعات را از بخش درواوه‌ی مرکزی دریافت کرده و در اختیار مصرف کنندگان یا ارائه‌دهنده‌گان خدمات مربوط به انرژی و منابع قرار دهد.

<sup>1</sup>Smart Metering System

<sup>2</sup>Central Communication Gateway

بخش درواوه‌ی ارتباطی مرکزی، خود به تنها یی متشکل از چندین بخش است؛ از جمله بخشی که بتواند با سخت‌افزار اندازه‌گیر ارتباط برقرار کرده و داده‌های اندازه‌گیری شده را از آن دریافت کند، بخشی که قادر است این اطلاعات جمع‌آوری شده را به کمک پروتکل‌های مورد استفاده در اینترنت اشیا به سمت ابر<sup>۱</sup> ارسال کند و در نهایت بخشی که بتواند این اطلاعات ذخیره کرده و در اختیار نرم‌افزار قرار دهد. لازم به ذکر است که ارزشمندی این اطلاعات ذخیره شده، همانطور که بالاتر نیز ذکر شد، بسیار بالاست و به همین علت می‌توان از آن در مصارف هوش تجاری<sup>۲</sup> و داده‌ی کلان<sup>۳</sup> نیز استفاده کرد. در شکل ۱-۱ می‌توان ارتباط بین این سه بخش را مشاهده کرد.



شکل ۱-۱ سیستم اندازه‌گیری هوشمند

### ۳-۱ درباره‌ی پروژه

در این پروژه تلاش شده است تا بخش درواوه‌ی ارتباطی مرکزی برای سامانه‌ی اندازه‌گیری هوشمند، طراحی و پیاده‌سازی شود. این درواوه وظیفه دارد در ابتدا داده‌های سخت‌افزار اندازه‌گیر را در ورودی دریافت کند و سپس به کمک پروتکل‌های اینترنت اشیا، این داده‌ها را بر روی بستر اینترنت ارسال کند تا در دسترس بخش نرم‌افزاری قرار گیرند.

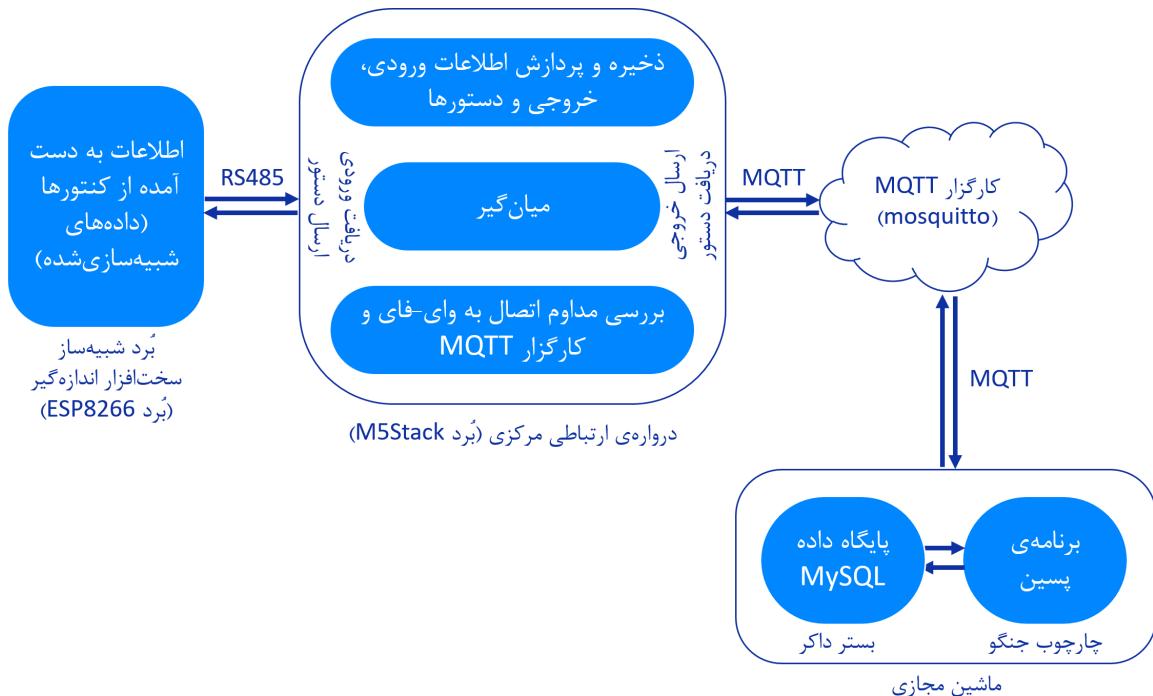
<sup>1</sup>Cloud

<sup>2</sup>Business Intelligence

<sup>3</sup>Big Data

همانطور که گفته شد تمرکز این پروژه بر روی بخش درواوهای مرکزی است و بخش‌های سخت‌افزار و نرم‌افزار خارج از محدوده‌ی این پروژه هستند. با این وجود برای بررسی صحت و درستی ارتباط با هر یک از این بخش‌ها، نیاز به ارتباطی هرچند به شکل شبیه‌سازی‌شده وجود داشته است. به همین سبب، بخشی به منظور تولید داده‌های شبیه‌سازی‌شده سخت‌افزار اندازه‌گیر به پروژه اضافه شده تا گامی در راستای تحقق هدف پروژه باشد.

در شکل ۲-۱ نمودار کاملی از بخش‌های مختلف پروژه و ارتباطات بین آن‌ها را مشاهده می‌کنید.



شکل ۲-۱ نمودار ارتباطات و بخش‌های مختلف پروژه

در ادامه‌ی این گزارش، در فصل دوم با پروتکل‌ها و همچنین سخت‌افزارها و فناوری‌های استفاده‌شده در پروژه آشنا می‌شویم و سپس در فصل سوم به تفصیل نحوه‌ی طراحی و پیاده‌سازی درواوهای مرکزی شرح داده خواهد شد. در پایان نیز نتایج به دست آمده از پروژه و همچنین پیشنهادهایی برای بهبود پروژه و گسترش آن را بیان می‌کنیم.

## فصل دوم

### مفاهیم پایه

برای آنکه درک مناسی از اهمیت وجودی درواره‌ی مرکزی در سیستم اندازه‌گیری هوشمند و نحوه ارتباط‌گیری آن با بخش‌های نرم‌افزاری و سخت‌افزاری ایجاد شود، نیاز است تا ابتدا برخی از مفاهیم پایه‌ای مرور شوند. در این فصل، به تشریح مفصل این موارد پرداخته شده است.

## ۱-۲ بردهای مورد استفاده در پروژه

پروژه از یک برد اصلی و یک برد جانبی به منظور تهیه و ارسال داده‌های شبیه‌سازی شده<sup>۱</sup> تشکیل شده است. در ادامه هر یک از این بردات به همراه ویژگی‌های اصلی‌شان شرح داده خواهد شد. برد اصلی بر پایه‌ی پردازنده‌ی ESP32 است که در بخش ۱-۱ به توضیح کامل‌تر وظیفه‌ی این برد در کنار ویژگی‌های دیگر سخت‌افزاری آن پرداخته شده است.

برد جانبی نیز که وظیفه‌ی رساندن اطلاعات شبیه‌سازی شده به برد اصلی را دارد بر روی پردازنده‌ی ESP8266 طراحی شده که شرح آن نیز در بخش ۲-۱ کامل آمده است.

## ۱-۱-۲ برد M5Stack با پردازنده‌ی ESP32

همانطور که در فصل اول گفته شد، هدف این پروژه این است که اطلاعات را از بخش سخت‌افزار کنتورها دریافت کند و سپس این اطلاعات را برای یک کارگزار MQTT<sup>۲</sup> (پروتکل MQTT از پروتکل‌های مهم در اینترنت اشیا که در بخش ۱-۲ به توضیح کامل آن پرداخته شده) ارسال کرده و در نهایت آن‌ها را در یک پایگاه داده<sup>۳</sup> ذخیره نماید. به همین منظور بردي انتخاب شده تا در ورودی خود این داده‌های سخت‌افزاری را دریافت کرده و سپس با پروتکل MQTT، این اطلاعات را ارسال نماید. بنابراین نیاز است تا برد مناسبی انتخاب گردد که هم از نظر توان پردازشی بتواند این میزان اطلاعات را مدیریت کند و هم امکان اتصال به اینترنت و ارسال داده را به راحتی فراهم آورد. از طرفی با نگاه به صنعت و هزینه‌ها، باید برد به گونه‌ای انتخاب شود تا برای مشتریانی که قصد استفاده از آن را دارند، صرفه‌ی اقتصادی داشته باشد.

بدین منظور بردات مختلفی از جمله بردات آردوینو، بردات رزبری‌پای، بردات سری ESP32

<sup>1</sup>Simulated

<sup>2</sup>MQTT Broker

<sup>3</sup>Database

مانند برد های M5Stack ساخت شرکتی با همین نام<sup>۱</sup> و برخی دیگر از برد ها بررسی شدند تا یک برد کارآمد و مقرون به صرفه برای این بخش از پروژه انتخاب شود.

### ۱-۱-۲ مقایسه و انتخاب برد اصلی پروژه

بردهای آردوینو<sup>۲</sup> در حوزه اینترنت اشیا صنعتی به طور گسترهای استفاده می شوند. این برد ها به خاطر قابلیت اطمینان، سهولت استفاده و جامعه ای برنامه نویسان و پشتیبان این برد ها شناخته شده اند. برد های آردوینو متن باز<sup>۳</sup> هستند؛ به این معنی که هم سخت افزار و هم نرم افزار آن ها قابل تغییر و سفارشی سازی<sup>۴</sup> هستند [۶].



شکل ۱-۲ نمونه ای از یک برد آردوینو

بردهای رزبری پای<sup>۵</sup> نیز در کاربردهای صنعتی اینترنت اشیا زیاد به چشم می خورند. آن ها به دلیل استحکام<sup>۶</sup>، مقرون به صرفه بودن و - همانند برد های آردوینو - داشتن پشتیبانی گسترده از طرف برنامه نویسان و دوستداران این برد ها شناخته شده اند. برد های رزبری پای با طیف گسترده ای از محصولات محاسباتی با توان های پردازشی متفاوت عرضه می شوند. آن ها با چندین زبان برنامه نویسی سازگار هستند

<sup>1</sup>M5Stack Technology Co.

<sup>2</sup>Arduino

<sup>3</sup>Open-Source

<sup>4</sup>Customization

<sup>5</sup>Raspberry Pi

<sup>6</sup>Robustness

و ویژگی‌های مختلفی مانند درگاه‌های HDMI<sup>۱</sup>، واحدهای پردازش گرافیکی<sup>۲</sup>، حافظه<sup>۳</sup>، درگاه‌های اترنت<sup>۴</sup>، اسلات کارت SD، پایه‌های ورودی و خروجی عمومی<sup>۵</sup> و موارد دیگر را ارائه می‌دهند[۷].

دسته‌های دیگری از بردّها نیز همانند بردّهای M5Stack هستند که آن‌ها نیز با هدف به کارگیری در صنعت اینترنت اشیا طراحی و توسعه داده شده‌اند. این دستگاه‌ها، مدل‌لار یا پیمانه‌ای<sup>۶</sup>، قابل انجام است. این پردازنده‌ها مقیاس‌پذیر<sup>۷</sup> و قابل حمل<sup>۸</sup> هستند که هسته‌ی پردازشی آن‌ها، پردازنده‌ی ESP32 است. این پردازنده‌ها به عنوان قلب هوشمند دستگاه‌ها عمل می‌کنند و از نظر عملکردی بسیار قوی و کارآمد هستند. علاوه بر این، بردّهای M5Stack به عنوان دستگاه‌های متن‌باز شناخته می‌شوند، به این معنی که کد منبع<sup>۹</sup> آن‌ها به طور عمومی در دسترس قرار دارد و توسعه‌دهندگان می‌توانند به راحتی کد را تغییر داده و به نیازهای خود سازگار کنند. این ویژگی مهم برای توسعه‌دهندگان است، زیرا آن‌ها را قادر می‌سازد تا در همه مراحل توسعه، از طراحی اولیه تا توسعه و پیاده‌سازی، به راحتی با بردّها کار کنند و به سرعت به حل مشکلات بپردازند[۸].

بردّهای آردوینو از لحاظ صرفه‌ی اقتصادی بسیار مناسب هستند اما در بسیاری از موارد، نمی‌توانند کارآمد باشند و توان پردازشی کافی برای این حجم از اطلاعات ورودی و خروجی را ندارند. بردّهای رزبری‌پای نیز از نظر هزینه، برای این پروژه توصیه نمی‌شوند؛ زیرا امکانات زیادی را به ما می‌دهند که ممکن است حتی یک بار هم استفاده نشوند و در این صورت تنها هزینه‌ی اضافی‌ای را بر روی مشتریان محصول خواهند گذاشت. اما بردّهای M5Stack به دلیل طراحی ساده و استفاده از پردازنده‌ی پرقدرت ESP32 در ساختار خود، هم از نظر هزینه بسیار به صرفه هستند و هم از طرفی، نیاز پردازشی ما را در این پروژه برآورده می‌کنند.

شایان ذکر است که یکی از نیازمندی‌های مهم در پروژه‌های اینترنت اشیا همچون این پروژه، توانایی

<sup>1</sup>High-Definition Multimedia Interface

<sup>2</sup>GPU

<sup>3</sup>RAM

<sup>4</sup>Ethernet Port

<sup>5</sup>GPIO

<sup>6</sup>Modular

<sup>7</sup>Stackable

<sup>8</sup>Scalable

<sup>9</sup>Portable

<sup>10</sup>Source Code

اتصال به شبکه‌ی بی‌سیم وای‌فای<sup>۱</sup> و اتصال بلوتوث<sup>۲</sup> است. از دیگر نیازهای پروژه می‌توان به پشتیبانی از حالات کم مصرف اشاره کرد؛ به این معنی که برد منتخب قادر باشد حالت‌های خواب عمیق<sup>۳</sup> و یا مصرف پایین<sup>۴</sup> را پشتیبانی کند. این ویژگی‌ها اجازه می‌دهند که دستگاه در دوره‌های زمانی که نیاز به ارتباط ندارد، به حالت خواب درآید و انرژی را مصرف نکند. این موضوع در میزان مصرف باتری و طول عمر برد تاثیر به سزایی خواهد داشت.

برد M5Stack به خوبی قادر است تمامی این موارد را پشتیبانی کند. با توجه به مقایسه‌های انجام شده و تحقیقات صورت‌گرفته، در نهایت بنا به دلایلی که پیشتر توضیح داده شد، برد M5Stack مدل پایه<sup>۵</sup> به عنوان برد اصلی این پروژه در نظر گرفته شد. در ادامه ویژگی‌ها و نکات مرتبط با این برد به تفصیل شرح داده شده است.

### ۲-۱-۱-۲ ویژگی‌ها و نکات مرتبط با برد منتخب

یکی از انواع بردهای M5Stack مدل پایه است. این برد، به عنوان یک سیستم مدولار، مأذول‌های مختلفی از جمله حسگرها، نمایشگر LCD<sup>۶</sup> و دکمه‌ها<sup>۷</sup> را پشتیبانی می‌کند. این امکان به توسعه دهنده‌گان اجازه می‌دهد تا بر اساس نیازهای خاص پروژه، مأذول‌ها را افروزه یا حذف کرده و سیستم را به راحتی تنظیم کنند.

در طراحی این برد از واحد وای‌فای به منظور اتصال برد به اینترنت استفاده شده است. همچنین این برد دارای یک پردازنده دو هسته‌ای با ۱۶ مگابایت فلش SPI است. این برد M5Stack از چندین زبان برنامه‌نویسی مانند آردوینو، زبان UIFlow با Blockly و Micropython پشتیبانی می‌کند. در شکل ۳-۲ می‌توانید تصویر این برد را مشاهده کنید.

همانطور که پیش‌تر نیز گفته شد، در این برد از هسته‌ی ESP32 استفاده شده است که یکی از پردازنده‌های قدرتمند در کارهای اینترنت اشیا به شمار می‌رود. در شکل ۳-۲، طرح‌واره‌ی<sup>۸</sup> درگاه‌های

<sup>1</sup> WiFi

<sup>2</sup> Bluetooth

<sup>3</sup> Deep Sleep Mode

<sup>4</sup> Low Power

<sup>5</sup> Basic Model: <https://docs.m5stack.com/en/core/basic>

<sup>6</sup> Liquid-Crystal Diode

<sup>7</sup> Buttons

<sup>8</sup> Schematic



شکل ۲-۲ برد M5Stack مدل ساده

متصل به این هسته‌ی پردازشی نشان داده شده است.

	GND	1	2	ADC1	GPIO35
	GND	3	4	ADC2	GPIO36
	GND	5	6	RESET	EN
GPIO23	MOSI	7	8	DAC0/AUDIO_L	GPIO25
GPIO19	MISO	9	10	DAC1/AUDIO_R	GPIO26
GPIO18	SCK	11	12	3.3V	
GPIO3	IO0/RXD1	13	14	IO1/TXD1	GPIO1
GPIO16	IO2/RXD2	15	16	IO3/TXD2	GPIO17
GPIO21	IO4/SDA	17	18	IO5/SCL	GPIO22
GPIO2	IO6	19	20	IO7	GPIO5
GPIO12	IO8/IIS_SCLK	21	22	IO9/IIS_WS	GPIO13
GPIO15	IO10/IIS_OUT	23	24	IO11/IIS_MCLK/BOOT	GPIO0
	HPWR	25	26	ADC0/IIS_IN	GPIO34
	HPWR	27	28	5V	
	HPWR	29	30	BATTERY	

شکل ۳-۲ طرح‌واره‌ی درگاه‌های متصل به هسته‌ی برد M5Stack

## ۲-۱-۲ برد ESP8266

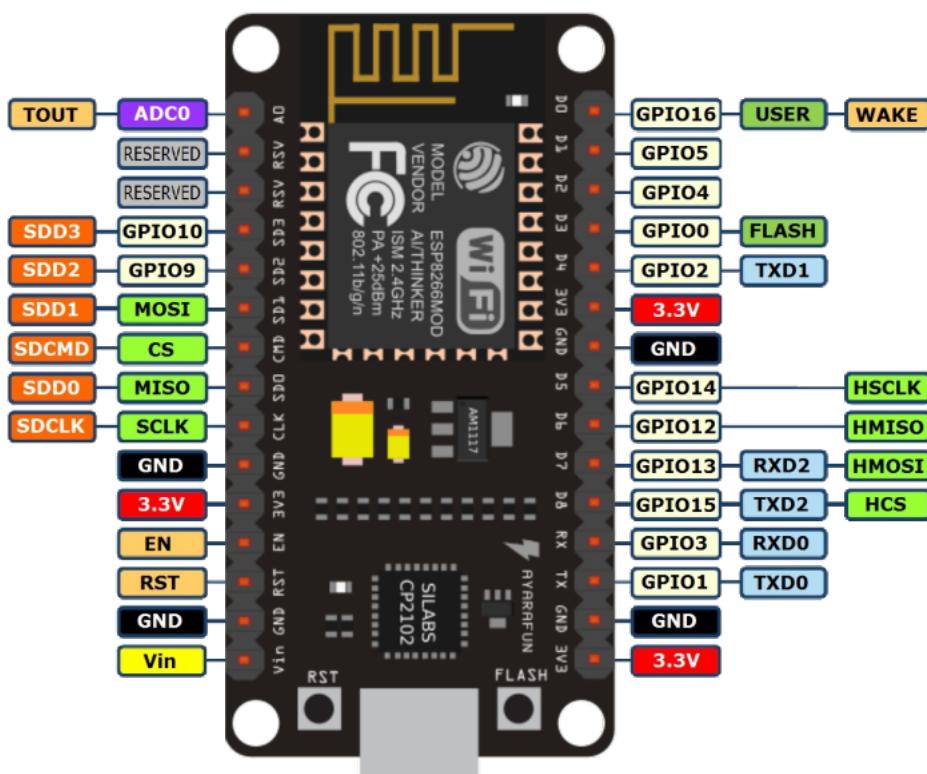
ما در این پروژه، برای اینکه بتوانیم داده‌های ورودی را به کمک پایه‌های<sup>1</sup> سخت‌افزاری دریافت کنیم و برد اصلی در شرایط مشابه به واقعیت بتواند عمل کند، نیاز داشتیم تا داده‌های شبیه‌سازی شده‌ای را به برد ورودی بدهیم. بدین منظور باید برد دیگری اضافه می‌شد تا این داده‌ها را شبیه‌سازی کرده و برای برد ما ارسال کند. برای این بخش از برد ESP8266 استفاده کردیم. از آنجا که این بخش از پروژه در

<sup>1</sup>Pin

واقعیت وجود ندارد و برد اصلی باید به کنتورها متصل گردد، نیازی به بهینه کردن قیمت و کارایی برد نبود اما با این وجود برای انتخاب شده که کارایی لازم را برای ما داشته باشد و در عین حال هزینه بر نباشد.

ESP8266 یک سیستم روی تراشه<sup>۱</sup> است که دارای ۱۶ پایه‌ی ورودی-خروجی همه منظوره، یک پایه‌ی ورودی آنالوگ، یک رابط UART، یک رابط SPI و یک رابط I2C است. این بردها با چندین زبان برنامه‌نویسی مانند آردوینو سازگار هستند و به راحتی با محیط توسعه یکپارچه‌ی آردوینو قابل برنامه‌ریزی هستند. برد های ESP8266 کم‌هزینه و ایده‌آل برای پروژه‌های اینترنت اشیا هستند که به اتصال وای‌فای و مصرف انرژی کم نیاز دارند.

در شکل ۴-۲ نمونه‌ای از این برد را به همراه طرح‌واره‌ی پایه‌های ورودی و خروجی آن مشاهده می‌کنید.



شکل ۴-۲ طرح‌واره‌ی درگاه‌های متصل به برد ESP8266

<sup>۱</sup>System-on-Chip (SoC)

### ۳-۱-۲ مژول فرستنده و گیرنده RS485

از مژول فرستنده-گیرنده RS485 هنگام انتقال داده بین میکروکنترلرهای استفاده می‌شود. این مژول فرستنده-گیرنده متشکل از مدار مجتمع<sup>۱</sup> Maxim MAX485 قادر است ارتباط سریال را در فواصل طولانی تا ۱۲۰۰ متر فراهم می‌کند. این مژول کارآمد قادر است داده‌ها را در هر دو جهت با حداکثر سرعت ۲/۵ مگابیت بر ثانیه انتقال دهد. این مژول که کار تبدیل TTL MAX485 به RS485 و برعکس را انجام می‌دهد به راحتی قادر است با میکروکنترلرهایی مانند آردوبینو، ESP8266، ESP32 و غیره ارتباط برقرار کند زیرا از سطوح منطقی  $\frac{3}{3}$  ولت استفاده می‌کند.<sup>[۹]</sup>

توضیحات کامل درباره استاندارد RS485 در بخش ۱-۳-۲ آمده است. در ادامه فهرست برخی از ویژگی‌های کلیدی مژول فرستنده و گیرنده RS485 را می‌آوریم:

۱- سیگنال‌دهی تفاضلی<sup>۲</sup> برای ایمنی بهتر در برابر اختلال<sup>۳</sup>

۲- پوشش دادن مسافت تا ۱۲۰۰ متر

۳- پشتیبانی از حداکثر سرعت انتقال داده‌ی ۲/۵ مگابیت بر ثانیه

۴- پشتیبانی از اتصال تا ۳۲ دستگاه بر روی گذرگاه مشترک<sup>۴</sup>

۵- ولتاژ کاری  $\frac{3}{3}$  ولت

۶- مصرف برق کم

تصویر این فرستنده و گیرنده در شکل ۵-۲ قابل مشاهده است.



شکل ۵-۲ تصویر مژول فرستنده و گیرنده RS485

<sup>۱</sup>Integrated Circuit (IC)

<sup>۲</sup>Differential Signalling

<sup>۳</sup>Noise

<sup>۴</sup>Bus

## ۲-۲ پروتکل‌ها و مفاهیم برای ارتباط با بخش نرم‌افزار

برای توضیح مفاهیم پایه در حوزه ارتباط با نرم‌افزار، ابتدا بهتر است مفاهیم مرتبط با پروتکل‌ها را توضیح دهیم. از پروتکل‌ها به عنوان یک زبان مشترک بین دستگاه‌های مختلف در اینترنت اشیا استفاده می‌شود. به همین دلیل یکی از مهم‌ترین و پایه‌ای‌ترین مواردی هستند که تسلط بر آن‌ها پیش از هر موضوع دیگری حائز اهمیت است.

### ۱-۲-۲ پروتکل MQTT

پروتکل MQTT، یک پروتکل ارتباطی است که برای برقراری ارتباطات بین دستگاه‌های مختلف اینترنت اشیا به کار می‌رود. این پروتکل به منظور تبادل پیام سبک در شبکه‌های با پهنای باند کم، تأخیر بالا یا غیرقابل اعتماد طراحی شده است که امکان ارتباط تجهیزات با منابع انسانی محدود با یکدیگر را به سادگی فراهم می‌سازد.<sup>[۱۰]</sup>

این پروتکل، مجموعه‌ای از قوانین است که نحوه انتشار<sup>۱</sup> و اشتراک<sup>۲</sup> داده‌ها را توسط دستگاه‌های اینترنت اشیا مشخص می‌کند. این پروتکل مبتنی بر رویداد<sup>۳</sup> است و دستگاه‌ها را با استفاده از الگوی ارتباطی انتشار-اشتراک به یکدیگر متصل می‌کند. فرستنده (ناشر<sup>۴</sup>) و گیرنده (مشترک<sup>۵</sup>) می‌توانند از طریق یک موضوع<sup>۶</sup> با هم ارتباط برقرار کنند و وظیفه‌ی برقراری این ارتباط بر عهده‌ی کارگزار MQTT خواهد بود. کارگزار MQTT تمام پیام‌های دریافتی را از فیلتر رد کرده و آن‌ها را به درستی بین مشترکین توزیع می‌کند.<sup>[۱۱]</sup>

### ۱-۲-۲-۱ کارگزار MQTT

کارگزار MQTT یک واسط بین دستگاه‌های اینترنت اشیا است که پروتکل MQTT را برای ارسال و دریافت پیام‌ها به کار می‌برد. استفاده از کارگزار MQTT، به دستگاه‌های مختلف کمک می‌کند تا بتوانند

<sup>1</sup>Publish

<sup>2</sup>Subscribe

<sup>3</sup>Event Driven

<sup>4</sup>Publisher

<sup>5</sup>Subscriber

<sup>6</sup>Topic

با یکدیگر ارتباط برقرار کنند و اطلاعات را با سرعت بالا و با حجم کم برای یکدیگر منتقل کنند<sup>[۱]</sup>. یک کارگزار در حقیقت بین دو یا چند کارخواه<sup>۱</sup> که یکی در نقش ناشر و بقیه در نقش مشترک هستند، ارتباط برقرار می‌کند. کارخواه‌ها به کارگزار اعلام می‌کنند که به چه موضوعاتی علاقه دارند و آماده‌ی دریافت داده بر روی آن‌ها هستند. کارگزار پیام‌ها را از کارخواه‌های مختلف دریافت می‌کند و آن‌ها را به کارخواه‌های دیگری که در همان موضوع مشترک شده‌اند، ارسال می‌کند. کارگزار به عنوان یک کارساز<sup>۲</sup> مرکزی عمل می‌کند و داده‌های منتشرشده توسط این کارساز مدیریت می‌شوند. در واقع، کارگزار مسئول توزیع و دریافت داده‌هاست. طرح‌واره‌ای از نحوه ارتباط بین دو کارخواه با یک کارگزار MQTT را می‌توانید در شکل ۶-۲ مشاهده کنید<sup>[۱]</sup>.



شکل ۶-۲ نحوه ارتباط کارخواه‌ها با یکدیگر به کمک کارگزار MQTT<sup>[۱]</sup>

راه حل جایگزینی به جای استفاده از کارگزار MQTT برای ارسال اطلاعات از یک برد به سمت کارساز ذخیره‌سازی اطلاعات وجود دارد. با استفاده از پروتکل HTTP، دستگاه مبدأ می‌تواند درخواست HTTP را به سمت کارساز ذخیره‌سازی اطلاعات ارسال کند و سپس کارساز مقصد می‌تواند پاسخ HTTP را به دستگاه مبدأ برگرداند و اطلاعات را ذخیره کند. اما پروتکل HTTP سربار زیادی برای کاربردهای اینترنت اشیا دارد و به همین دلیل پرهزینه است. پس راه حل کارگزار MQTT گزینه‌ی بهتری برای ارسال اطلاعات خواهد بود. در ادامه برخی از مزایای استفاده از کارگزار MQTT در مقایسه با پروتکل HTTP آورده شده است:

۱- قابلیت شبکه کردن چند دستگاه روی یک گذرگاه مشترک

۲- قابلیت تحمل اختلال بالا

۳- قابلیت استفاده در فواصل طولانی

<sup>1</sup>Client

<sup>2</sup>Server

#### ۴- سرعت بالا در ارسال و دریافت داده‌های مناسب اینترنت اشیا

##### ۲-۱-۲-۲ مقایسه‌ی کارگزارهای MQTT

در این بخش به مقایسه‌ی کارگزارهای MQTT می‌پردازیم. البته تعداد این کارگزارها زیاد است و مقایسه‌ی تمامی آن‌ها از محدوده‌ی این پژوهه خارج است. اما به هر حال در روند پژوهه، مقایسه و انتخاب یک کارگزار MQTT بین گزینه‌های موجود، اجتناب‌ناپذیر بوده است. در بین کارگزارهای مختلف، HiveMQ و Mosquitto دو تا از کارگزارهای مهم هستند که در صنعت و تحقیقات نیز بسیار به چشم می‌خورند و شناخته شده هستند [۱۲، ۱۳].

با استناد به مقایسه‌های انجام گرفته و هم‌چنین نیاز پژوهه، کارگزار پیام Mosquitto برای پیشبرد اهداف این پژوهه انتخاب شده است. در ادامه برخی از نکات مربوط به این کارگزار و نتیجه‌ی مقایسه‌های انجام گرفته آمده است.

یک پیاده‌سازی متن‌باز از پروتکل MQTT است که توسط بنیاد Eclipse<sup>۱</sup> توسعه داده شده است. برای ارتباط با Mosquitto، می‌توان از دستورهای ”mosquitto\_sub“ و ”mosquitto\_pub“ استفاده کرد. دستور ”mosquitto\_pub“ برای انتشار پیام‌ها به موضوع‌های MQTT و ”mosquitto\_sub“ برای مشاهده پیام‌های ارسال شده از موضوع‌های MQTT به کار می‌رود [۱۴].

Mosquitto با کارگزارهای دیگر مانند HiveMQ<sup>۲</sup> و EMQX<sup>۳</sup> قابل مقایسه است. در حال حاضر، Mosquitto یکی از پرطرفدارترین پیاده‌سازی‌های MQTT است. این کارگزار پیام<sup>۴</sup>، به دلیل سبک و قابل استفاده بودن در تمام دستگاه‌ها، محبوبیت زیادی در بین کاربران پیدا کرده است [۱۳، ۱۴].

برتری‌های Mosquitto عبارتند از:

۱- سبک بودن و پیاده‌سازی شده با زبان C

۲- قابل استفاده بودن در تمام دستگاه‌ها

۳- سادگی کار و یادگیری آسان

<sup>1</sup>Eclipse Foundation

<sup>2</sup><https://www.hivemq.com/>

<sup>3</sup><https://www.emqx.io/>

<sup>4</sup>Message Broker

#### ۴- متن باز بودن و دسترسی رایگان

### ۲-۲-۲ زیرساخت ارتباطی با بخش نرم افزار

در این بخش قصد داریم به توضیح مفاهیم و فناوری هایی که برای ایجاد ارتباط با بخش نرم افزار در این پروژه به کار گرفته ایم، بپردازیم.

برای ارتباط با بخش نرم افزار، از یک پایگاه داده از نوع MySQL استفاده کردیم. این پایگاه داده به ما امکان می دهد میزان مصرف انرژی و منابع را ذخیره کرده و در اختیار توسعه دهنده کان نرم افزار قرار دهیم. برای سادگی راه اندازی و بهبود مدیریت، از مفهوم ماشین مجازی<sup>۱</sup> استفاده کردیم. سپس با راه اندازی سرویس داکر<sup>۲</sup> بر روی این ماشین مجازی موفق شدیم پایگاه داده را در یک محیط کاملاً منزوع<sup>۳</sup> مستقر<sup>۴</sup> کنیم. در نهایت با اجرا کردن برنامه‌ی پسین به کمک چارچوب نرم افزاری جنگو<sup>۵</sup>، هدف دریافت اطلاعات از کارگزار MQTT و ذخیره‌ی آنها در پایگاه داده به طور کامل محقق شد. در ادامه‌ی این بخش، جزئیات بیشتری در مورد هر یک از این فناوری‌های استفاده شده، شرح داده خواهد شد.

### ۱-۲-۲-۲ زیرساخت ماشین مجازی

ماشین مجازی یک فناوری است که امکان اجرای چندین سیستم عامل<sup>۶</sup> مستقل را روی یک سخت افزار فیزیکی فراهم می کند. با استفاده از ماشین مجازی، می توانیم پایگاه داده را بر روی یک محیط مجازی ایجاد کرده و بدون نیاز به راه اندازی پایگاه داده بر روی یک کارساز جداگانه، آن را بر روی سیستم شخصی خود اجرا کنیم. این کار به ما این امکان را می دهد که پایگاه داده را در یک محیط کنترل شده و منزوع اجرا کرده و همچنین محیطی مشابه با کارساز را برای توسعه دهنده کان نرم افزار ایجاد کنیم.

برای آماده کردن ماشین مجازی از زیرسیستم ویندوز برای لینوکس (WSL)<sup>۷</sup> استفاده کردیم. WSL اولین بار در سال ۲۰۱۶ میلادی منتشر شد. WSL2 یک فناوری مجازی سازی سبک است که بر روی

<sup>۱</sup>Virtual Machine

<sup>۲</sup>Docker

<sup>۳</sup>isolated

<sup>۴</sup>Deploy

<sup>۵</sup>Django Software Framework

<sup>۶</sup>Operating System

<sup>۷</sup>Windows Subsystem for Linux

سیستم عامل ویندوز<sup>۱</sup>، نصب و اجرا می‌شود و به کاربران این سیستم عامل، امکان اجرای باینری سیستم عامل لینوکس<sup>۲</sup> را می‌دهد. استفاده از WSL2 مزایایی را همراه خود دارد؛ از جمله دسترسی به ترمینال<sup>۳</sup> لینوکس در ویندوز، توسعه برنامه‌های کاربردی چندسکویی<sup>۴</sup> و مدیریت زیرساخت‌های فناوری اطلاعات بدون نیاز به خروج از ویندوز[۱۵].

پس از نصب و راهاندازی WSL2 بر روی سیستم شخصی، به نصب ماشین مجازی Ubuntu 22.04.2 LTS پرداختیم. این ماشین مجازی در حقیقت در نقش کارساز مرکزی در این پروژه عمل می‌کند و همان زیرساخت مورد نیاز برای راهاندازی برنامه‌ی پسین و پایگاه داده می‌باشد.

## ۲-۲-۲-۲ سرویس داکر

در این پروژه از سرویس داکر به منظور راهاندازی پایگاه داده استفاده شده است. داکر یک سکوی<sup>۵</sup> متن باز برای توسعه، جابه‌جایی و اجرای برنامه‌ها است. به کمک داکر می‌توان برنامه‌های کاربردی را از زیرساخت جدا کرد تا بتوان نرم‌افزار را به سرعت تحویل داد. با بهره‌گیری از روش‌شناسی‌های<sup>۶</sup> داکر برای ارسال، آزمایش و استقرار<sup>۷</sup> برنامه، می‌توان تاخیر بین نوشتن برنامه و اجرای آن را به میزان قابل توجهی کاهش داد[۱۶].

## ۳-۲-۲-۲ چارچوب نرم‌افزاری جنگو

جنگو یک چارچوب توسعه وب با زبان پایتون<sup>۸</sup> است که ما را در روند توسعه سریع و طراحی شیوه‌مند<sup>۹</sup> یاری می‌کند. این چارچوب توسط توسعه‌دهندگان ماهر طراحی شده است و بسیاری از دشواری‌های توسعه وب را برای برنامه‌نویسان ساده می‌کند. این چارچوب رایگان و متن باز است. از جمله ویژگی‌های مهم دیگر این چارچوب می‌توان به سریع بودن، امن بودن و قابلیت مقیاس‌پذیری بالای آن اشاره کرد[۱۷].

<sup>1</sup>Windows

<sup>2</sup>Linux

<sup>3</sup>Terminal

<sup>4</sup>Cross-platform

<sup>5</sup>Platform

<sup>6</sup>Methodology

<sup>7</sup>Deployment

<sup>8</sup>Python

<sup>9</sup>Pragmatic

## ۳-۲ پروتکل‌ها و مفاهیم برای ارتباط با بخش سخت‌افزار

در این بخش برای درک بهتر نحوه اتصال برد شبیه‌ساز به برد اصلی، به توضیح پروتکل‌ها و کتابخانه‌های مورد استفاده در این اتصال خواهیم پرداخت. اتصال به کمک پروتکل Modbus بر روی استاندارد لایه‌ی فیزیکی RS485 برقرار شده است. Modbus شکل و قوانین تبادل داده بین دستگاه‌ها را مشخص می‌کند؛ در حالی که RS485 ویژگی‌های الکتریکی سیم‌های مورد استفاده برای انتقال داده را مشخص می‌کند. Modbus را می‌توان با لایه‌های فیزیکی مختلف مانند RS485، RS232، Ethernet و غیره به کار برد. در ادامه RS485 را می‌توان با پروتکل‌های ارتباطی مختلف مانند Modbus، Profibus و غیره به کار برد. در ادامه ویژگی‌های دقیق‌تر هر یک را توضیح خواهیم داد.

### ۱-۳-۲ استاندارد RS485

RS485 یک گذرگاه ارتباطی سریال ناهمzman<sup>۱</sup> است که به عنوان TIA-485<sup>۲</sup> یا EIA-485 استاندارد شده است. RS مخفف «استاندارد توصیه شده»<sup>۳</sup> است که مجموعه‌ای از استانداردهای رابط ارتباطی است که توسط اتحادیه صنایع الکترونیک<sup>۴</sup> نگهداری می‌شود.

در این استاندارد فیزیکی از یک کابل متعادل<sup>۵</sup> دارای دو سیم به صورت جفت به همتابیده استفاده می‌شود. این کابل می‌تواند تا ۱۲۰۰ متر گسترش یابد و حداکثر ۳۲ دستگاه می‌توانند به همان گذرگاه مشترک به صورت نیمه دوطرفه<sup>۶</sup> متصل شوند. به این معنی که به هر دستگاه در هر لحظه از زمان اجازه‌ی ارتباط فقط در یک جهت داده می‌شود.

در شکل ۷-۲ نمونه‌ای از گذرگاه RS485، با چهار دستگاه آورده شده است. همانطور که می‌بینید دستگاه‌ها به صورت سریالی به هم متصل شده‌اند. حتی اگر پیکربندی‌های دیگری امکان‌پذیر باشند، این پیکربندی بهترین عملکرد را دارد.<sup>۹</sup>

داده‌ها به صورت سریال از طریق کابل ارسال می‌شوند و سیگنال الکتریکی ارسالی ماهیت تفاضلی خواهد داشت. برای درک بهتر ماهیت تفاضلی دو سیگنال الف و ب را تصور کنید. سیگنال‌های الف و ب

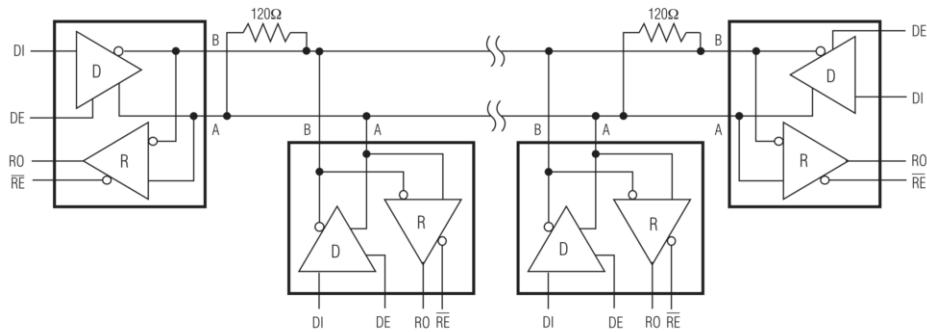
<sup>1</sup> Asynchronous Serial Communication Bus

<sup>2</sup> Recommended Standard

<sup>3</sup> Electronic Industries Alliance

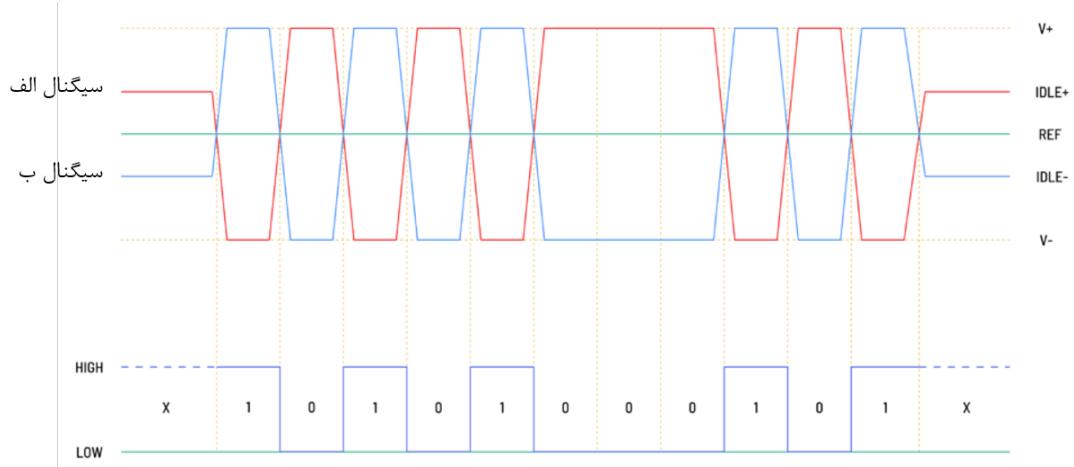
<sup>4</sup> Balanced

<sup>5</sup> Half-Duplex



شکل ۷-۲ شبکه‌ی نیمه دوطرفه RS485

یک خط داده‌ی تفاضلی واحد برای رابط RS485 را تشکیل می‌دهند. در طول ارتباط فعال، ولتاژهای خطوط الف و ب از نظر قطبیت<sup>۱</sup> مخالف خواهند بود. به عنوان مثال، اگر سیگنال الف در سطح ولتاژ مثبت باشد، سیگنال ب در سطح منفی خواهد بود و بالعکس. هر دو سیگنال الف و ب نمی‌توانند در طول ارتباط فعال در یک حالت یا سطح ولتاژ باشند، زیرا هدف استفاده از سیگنال‌دهی تفاضلی را از بین می‌برد. در شکل ۸-۲ نیز می‌توانید نمودار شکل موج RS485 را با توجه به سطح ولتاژ این دو سیگنال مشاهده کنید [۹].



شکل ۸-۲ شبکه‌ی RS485

<sup>۱</sup>Polarity

## ۲-۳-۲ Modbus پروتکل

پروتکل Modbus یک پروتکل ارتباطی است که در ابتدا توسط Modicon در سال ۱۹۷۹ برای استفاده با کنترل کننده‌های منطقی قابل برنامه‌ریزی (PLC)<sup>۱</sup> توسعه یافت. از آن زمان به یک استاندارد پرکاربرد برای اتصال دستگاه‌های الکترونیکی صنعتی تبدیل شده است. Modbus به دو نوع واحد ترمینال از راه دور (RTU)<sup>۲</sup> و ASCII تقسیم می‌شود. در پروتکل RTU، داده‌ها به صورت باینری ارسال می‌شوند؛ در حالی که در پروتکل ASCII، داده‌ها به صورت کاراکتر ارسال می‌شوند.<sup>[۱۸، ۱۹]</sup>

برای کار با پروتکل Modbus معمولاً از کتابخانه‌های آماده استفاده می‌شود. به عنوان مثال، کتابخانه modbus-esp8266 که برای ESP8266/ESP32 توسط الکساندر امیلیانوف<sup>۳</sup> نوشته شده یک کتابخانه Modbus بسیار ساده و قابل استفاده است که از آن برای ایجاد ارتباط بین برد اصلی و برد شبیه‌ساز در این پروژه استفاده کردۀ‌ایم.<sup>[۲۰]</sup>

<sup>۱</sup>Programmable Logic Controllers

<sup>۲</sup>Remote Terminal Unit

<sup>۳</sup>Alexander Emelianov

## فصل سوم

### طراحی و پیاده‌سازی درواوه‌ی مرکزی

در این فصل، به تشریح مفصل نحوه طراحی و پیاده‌سازی درواوه‌ی مرکزی ارتباطی پرداخته شده است. پروژه‌ی درواوه‌ی ارتباطی مرکزی برای سیستم اندازه‌گیری هوشمند مصارف ساختمانی، به صورت یک میان‌افزار بین دو بخش سخت‌افزاری و نرم‌افزاری واقع شده و کار برقراری ارتباط بین این دو بخش را انجام می‌دهد. در این پروژه داده‌های حسگرها و کنتورها از طریق پروتکل MQTT از این میان‌افزار به کارگزار ارسال می‌شوند و سپس از کارگزار به سمت یک ماشین مجازی که روی آن پایگاه داده نیز سوار است و به عنوان کارساز ذخیره‌سازی در این پروژه در نظر گرفته شده، ارسال می‌شود.

به دلیل تنوع زیرساختی در بخش‌های مختلف این پروژه، روند پیاده‌سازی کامل آن به سه بخش کلی تقسیم شده است. بخش اول توضیح می‌دهد که چگونه ارتباط با کارگزار MQTT از طریق برد اصلی استفاده شده در پروژه، یعنی برد M5Stack برقرار می‌گردد. در بخش دوم مشخص می‌شود که این برد اطلاعات شبیه‌سازی شده و ورودی خود را به چه صورت از برد شبیه‌ساز ESP8266 دریافت می‌کند و در انتهای روند ارسال اطلاعات از کارگزار MQTT به سمت ماشین مجازی به جهت ذخیره‌سازی اطلاعات توضیح داده خواهد شد.

### ۱-۳ اتصال برد اصلی به کارگزار MQTT

هدف و وظیفه‌ی بخش اصلی پروژه‌ی درواوه‌ی ارتباطی مرکزی که ایجاد ارتباط بین کنتورها و پایگاه داده است، بر عهده‌ی برد M5Stack است. این برد باید در ابتدا بتواند داده‌های ورودی را بخواند، آن‌ها را پردازش کند و سپس با کمک پروتکل MQTT برای ماشین مجازی که پایگاه داده بر روی آن مستقر است، ارسال نماید.

پیش از هرچیز لازم است تا خود برد را راه‌اندازی کنیم. برای این کار از کتابخانه‌ی مخصوص خود استفاده کردیم و آماده‌سازی‌های لازم را جهت کار با برد انجام دادیم. همان‌طور که در قطعه کد آورده شده در شکل ۱-۳ مشاهده می‌کنید، به کمک تابع `begin()` مخصوص این کتابخانه، برد راه‌اندازی می‌شود و سپس می‌توانیم از امکانات مختلف آن همچون میزان برق یا باتری برد<sup>۱</sup> یا نمایشگر استفاده کنیم.

حال برد آماده‌ی استفاده، باید ابتدا دسترسی به اینترنت داشته باشد تا بتواند کار ارسال داده را پیش

<sup>۱</sup>Power

```

1 #include <M5Stack.h>
2
3 // Board's unique ID
4 String m5stack_id = "";
5
6 void setup() {
7     // Set software serial baud to 115200;
8     Serial.begin(115200);
9     ...
10    // prepare board
11    M5.begin();
12    M5.Power.begin();
13    M5.Lcd.setTextSize(2);
14    M5.Lcd.println("Welcome :)");
15    ...
16 }
```

شکل ۱-۳ قطعه کد مربوط به راهاندازی برد M5Stack

ببرد. برای ایجاد این دسترسی از کتابخانه WiFi.h استفاده شده است. به کمک این کتابخانه ما می‌توانیم یک موجودیت WiFiClient ایجاد کنیم و سپس با داشتن نام و رمزعبور یک سرویس وای‌فای، به آن متصل شویم. قطعه کد مربوط به این بخش را می‌توانید در شکل ۲-۳ مشاهده کنید.

```

1 #include <WiFi.h>
2
3 // WiFi (Replace with your network credentials)
4 const char *ssid = "*****"; // Enter your WiFi name
5 const char *password = "*****"; // Enter WiFi password
6 WiFiClient espClient;
7
8 void setup() {
9     ...
10    // connecting to a WiFi network
11    WiFiInit();
12    m5stack_id = String(WiFi.macAddress());
13    ...
14 }
```

شکل ۲-۳ قطعه کد مربوط به اتصال برد به وای‌فای

یکی از چالش‌های این بخش این است که برد باید همواره دسترسی به اینترنت داشته باشد تا هر زمان که لازم بود داده‌ی جدیدی را برای کارگزار بفرستد، دچار مشکل نشود. به دلایل مختلفی از جمله، قطعی خود سرویس وای‌فای، آنتن‌دهی ضعیف یا دیگر مشکلاتی از این دست، برد ممکن است برای لحظاتی دسترسی خود به اینترنت را از دست بدهد. در این صورت قطعه کد WiFi.begin() نمی‌تواند دوباره برد را به وای‌فای متصل کند و تنها برای بار اول قابل استفاده است. به منظور حل این مشکل از مفهومی

به نام رویداد<sup>۱</sup> در کد استفاده کردیم. رویدادها به ما این امکان را می‌دهند که به محض مشاهده‌ی تغییر، عملکرد لازم را اجرا کنیم. در شکل ۳-۳ در تابع WiFiInit() مشاهده می‌کنید که از سه رویداد مختلف استفاده شده است تا این مشکل را برطرف کند. رویداد WiFiStationConnected به محض اینکه برد به اینترنت متصل شود، پیام اتصال را چاپ می‌کند. رویداد WiFiGotIP زمانی اجرا می‌شود که آی‌پی<sup>۲</sup> به برد موردنظر اختصاص داده شود و این آی‌پی را چاپ می‌کند. رویداد سوم یا WiFiStationDisconnected هر زمان که برد ما از اینترنت قطع شود، اجرا می‌شود و به کمک WiFi.begin() مجدد اتصال را برقرار خواهد کرد.

```

1 void WiFiInit() {
2     // delete old config
3     WiFi.disconnect(true);
4
5     delay(1000);
6
7     // handle different Wi-Fi events
8     WiFi.onEvent(WiFiStationConnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_CONNECTED);
9     WiFi.onEvent(WiFiGotIP, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_GOT_IP);
10    WiFi.onEvent(WiFiStationDisconnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_DISCONNECTED);
11
12    WiFi.begin(ssid, password);
13    Serial.println("Connecting to WiFi...");
14 }
```

شکل ۳-۳ قطعه کد رویدادهای اتصال به وای-فای

پس از اتصال به اینترنت باید اتصال برد به کارگزار MQTT را راهاندازی کنیم و همانند اتصال به اینترنت مطمئن شویم که این اتصال اگر به هر دلیل قطع شد، می‌تواند دوباره برقرار شود. در این بخش از کتابخانه PubSubClient استفاده کردہ‌ایم. همانطور که در فصل دوم اشاره شد، کارگزاری که در این پروژه انتخاب شده است، کارگزار Mosquitto نام دارد. این کارگزار، یک کارساز به صورت آزمایشی راهاندازی کرده که افراد می‌توانند برای کارهای تحقیقاتی و با حجم کار کم به راحتی به آن متصل شوند. به علاوه این کارساز امکان اتصال با احراز هویت را نیز بر روی درگاه<sup>۳</sup> ۱۸۸۴ loop() معرفی شده. نحوه اتصال به این کارساز در کد را می‌توانید در شکل ۴-۳ مشاهده کنید.

همانطور که مشاهده می‌کنید، برای اتصال مجدد به کارگزار تابع reconnectBroker() تعریف شده است که در تابع loop() برد نیز هر زمان که ارتباط برد با کارگزار دچار اختلال شود، دوباره این تابع اجرا

<sup>1</sup>Event

<sup>2</sup>IP

<sup>3</sup>Port

```

1 #include <PubSubClient.h>
2
3 // MQTT Broker
4 const char *mqtt_broker = "test.mosquitto.org";
5 const int mqtt_port = 1884;
6 const char *mqtt_username = "rw";
7 const char *mqtt_password = "readwrite";
8
9 void setup() {
10     ...
11     // connecting to a mqtt broker
12     client.setServer(mqtt_broker, mqtt_port);
13     client.setCallback(callback);
14     reconnectBroker();
15     ...
16 }
```

شکل ۴-۳ قطعه کد اتصال به کارگزار MQTT

خواهد شد.

از آنجایی که چندین داده‌ی مختلف قرار است به سمت کارگزار ارسال شوند، ما نیاز به موضوع‌های مختلفی برای این هدف داریم. موضوع‌هایی که در کد استفاده شده‌اند به شرح زیر است که در آن بخش <Board ID> بعدا با آدرس MAC<sup>1</sup> مربوط به برد مشخص خواهد شد تا هر برد، موضوعات اختصاصی مربوط به خود را داشته باشد. این بخش در حقیقت پروتکلی بین دروازه‌ی مرکزی و بخش کارسازی است که می‌خواهد این اطلاعات را از کارگزار دریافت کرده و در پایگاه داده ذخیره کند.

AUTSmartMeteringSystem/gas/<Board ID>/consumption •

AUTSmartMeteringSystem/water/<Board ID>/consumption •

AUTSmartMeteringSystem/power/<Board ID>/consumption •

AUTSmartMeteringSystem/battery/<Board ID>/remainingPercentage •

AUTSmartMeteringSystem/command/<Board ID>/commandText •

یکی دیگر از نیازهای مهم پروژه این است که بتواند داده‌های ورودی را در یک میان‌گیر<sup>2</sup> ذخیره کند. این کار قابلیت اطمینان برد را بالاتر می‌برد. زیرا اگر به هر دلیلی برد نتواند برای مدتی اطلاعات را به

<sup>1</sup>Media Access Control Address (MAC Address)

<sup>2</sup>Buffer

سمت کارگزار ارسال کند، این داده‌ها از دست نخواهند رفت و در میان‌گیر ذخیره می‌شوند تا زمانی که مجدد به سمت کارگزار ارسال شوند.

در نهایت اطلاعاتی که از طریق ارتباط RS485 با پروتکل Modbus از ورودی برد M5Stack خوانده می‌شود بر روی موضوعات ذکر شده در بالا، به سمت کارگزار به صورت دوره‌ای ارسال خواهد شد. طول دوره برای ارسال هر یک از این داده‌ها در کد قابل تنظیم است. تابع receive\_and\_send\_data() کار دریافت داده از برد شبیه‌ساز و ارسال آن‌ها را انجام می‌دهد. برای ارسال داده، تابع send\_topic\_data() کار اطلاعات ورودی را به شکل json<sup>1</sup> در می‌آورد و آن را به موضوع مربوط می‌فرستد.

### ۲-۳ اتصال برد شبیه‌ساز به برد اصلی

در این بخش قصد داریم تا نحوه ارسال داده از سمت برد شبیه‌ساز به برد اصلی را شرح دهیم. این بخش در واقعیت و استفاده‌ی صنعتی می‌باشد با اتصال کنتور به برد اصلی، جایگزین شود. در آن صورت نیز نحوه ارسال و دریافت داده، همچنان با استاندارد RS485 خواهد بود، اما نحوه دقیق اتصال کنتور به این استاندارد فیزیکی با اتصال برد شبیه‌ساز متفاوت خواهد بود. از آنجا که بخش سخت‌افزاری و اتصالات کنتور در محدوده‌ی این پژوهه نمی‌گنجد، بنابراین تنها اتصال برد شبیه‌ساز ESP8266 به این استاندارد کافی خواهد بود.

ما از کتابخانه ModbusRTU برای هر دو برد ESP8266 و M5Stack استفاده کردیم. این کتابخانه، از توابعی هم برای دستگاه ارباب<sup>2</sup> و هم برای دستگاه‌های برد، برخوردار است و می‌تواند پاسخ‌گوی نیاز هر دو نوع ارتباطی باشد. دستگاه ارباب در این اتصال بهتر است دستگاهی باشد که در اکثر موارد داده ارسال می‌کند تا دریافت کند؛ به این منظور که دستگاه ارباب مدام لازم نباشد منتظر دریافت و خواندن پیام بماند. در پژوهه‌ی ما، این موضوع برای برد ESP8266 صدق می‌کند، زیرا این برد باید به شکل مداوم میزان مصرف آب، برق و گاز را به سمت برد اصلی ارسال نماید و تنها در صورتی که دستوری از سمت برد اصلی بیاید، نیاز به خواندن از پروتکل ارتباطی Modbus را خواهد داشت.

در فصل دوم به تفصیل، پروتکل RS485 شرح داده شد و گفته شد که این ارتباط از دو سیم جفت به هم تابیده تشکیل شده است. برد های فرستنده و گیرنده RS485 که در پژوهه استفاده شده‌اند، کار

<sup>1</sup>Master

<sup>2</sup>Slave

## فصل سوم: طراحی و پیاده‌سازی دروازه‌ی مرکزی

ارسال داده بر روی این استاندارد فیزیکی را انجام می‌دهند. اما برای فعال‌سازی این ارتباط ما به سه اتصال بین برد ESP8266 و برد مبدل RS485 نیاز داریم. این سه ارتباط، شامل پایه‌ی DE\\_RE به منظور مشخص کردن جهت ارسال/دریافت داده، پایه‌ی RX به منظور خواندن داده‌ی دریافتی و پایه‌ی TX به منظور ارسال داده تعریف شده‌اند.

در جدول ۱-۳، به طور کامل تمامی پایه‌های ورودی و خروجی مبدل فرستنده و گیرنده RS485 آورده شده است.

جدول ۱-۳ جدول پایه‌های مأذول فرستنده و گیرنده RS485

نام پایه	شرح
VCC	این پایه منبع تغذیه است. با ولتاژ ۵ ولت وصل شده است که مأذول را تغذیه می‌کند.
A	این پایه‌ی ورودی و خروجی راهانداز گیرنده غیروارون‌ساز <sup>*</sup> است. در مأذول دیگر با A متصل است.
B	این پایه‌ی ورودی گیرنده وارون‌ساز <sup>**</sup> و خروجی راهانداز است. به B در مأذول دیگر متصل است.
GND	این پایه‌ی GND است. با زمینه مشترک مرتبط است.
RO	این پایه خروجی گیرنده است. این پایه به پایه‌ی RX میکروکنترلر متصل می‌شود.
RE	این پایه جهت فعال کردن خروجی گیرنده است. برای فعال کردن، در حالت LOW تنظیم می‌شود.
DE	این پایه جهت فعال کردن خروجی راهانداز است. برای فعال کردن، در حالت HIGH تنظیم شده است.
DI	این پایه‌ی ورودی راهانداز است. این پایه به پایه‌ی TX میکروکنترلر متصل می‌شود.

driver \*

non-inverting \*\*

inverting \*\*\*

از ۱-۳ برای بستن ارتباطات بین بردها استفاده کرده‌ایم. ما برای این پروژه به اجزای زیر نیاز داریم:

۱- یک برد ESP8266 NodeMCU که کد برد شبیه‌ساز بر روی آن بارگزاری شده است.

۲- یک برد M5Stack که کد برد اصلی بر روی آن بارگزاری شده است.

۳- دو مأذول فرستنده و گیرنده RS485

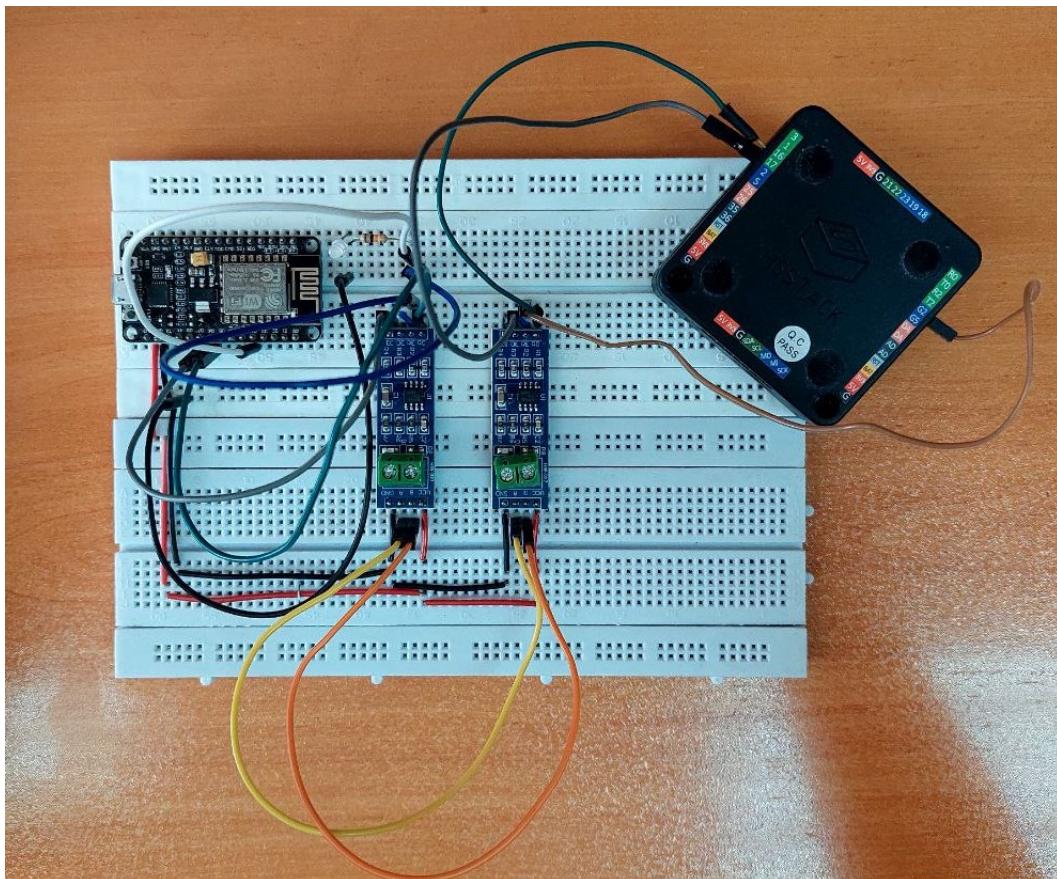
۴- یک مقاومت

۵- یک چراغ LED

۶- سیم‌هایی برای اتصال

۷- نمونه تابلویی<sup>۱</sup>

برای اتصال صحیح همه دستگاه‌ها با موفقیت، اتصالات را به دقت دنبال کنید.<sup>۲</sup>



شکل ۳-۵ تصویر کامل اتصالات

ابتدا اتصالات بین ارباب ESP8266 را توضیح خواهیم داد. ESP8266 به یک LED و یک ماژول RS485 متصل می‌شود. در جدول ۲-۳ می‌توانید اتصالات بین NodeMCU و ماژول فرستنده گیرنده را مشاهده کنید.

طبق این جدول، پایه‌ی VCC RS485 به پایه‌ی Vin ESP8266 و هر دو پایه‌ی GND بردها به یکدیگر وصل می‌شوند. پایه RO به پایه‌ی سریال RX ESP8266 متصل خواهد شد. به همین ترتیب، پایه

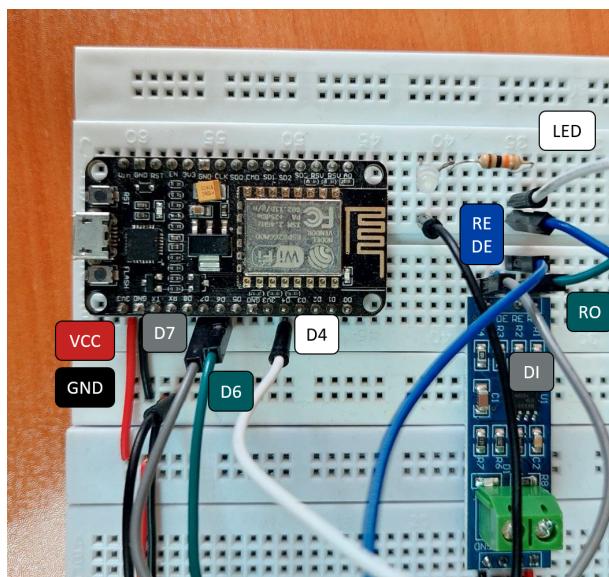
<sup>1</sup>Breadboard

<sup>2</sup>بخش مدار مربوط به LED و مقاومت تنها برای چک کردن ارسال داده از سمت برد شبیه‌ساز می‌باشد و لزومی به وجود آن نیست.

جدول ۲-۳ جدول اتصالات برد ESP8266 به مازول فرستنده و گیرنده RS485

RS485 ماژول فرستنده گیرنده	ESP8266
VCC	Vin
GND	GND
RO	D6 (GPIO12) یا پایه‌ی RX
RE و DE	D4 (GPIO2)
DI	D7 (GPIO13) یا پایه‌ی TX

DI به پایه‌ی سریال TX ESP8266 متصل خواهد شد. پایه‌های RE و DE با هر پایه‌ی ورودی-خروجی عام منظوره‌ی<sup>۱</sup> برد به هم متصل خواهند شد. برای اتصال این دو پایه از GPIO2 استفاده کردیده‌ایم. نحوه‌ی بستن این اتصالات را در شکل ۶-۳ نیز می‌توانید ببینید.

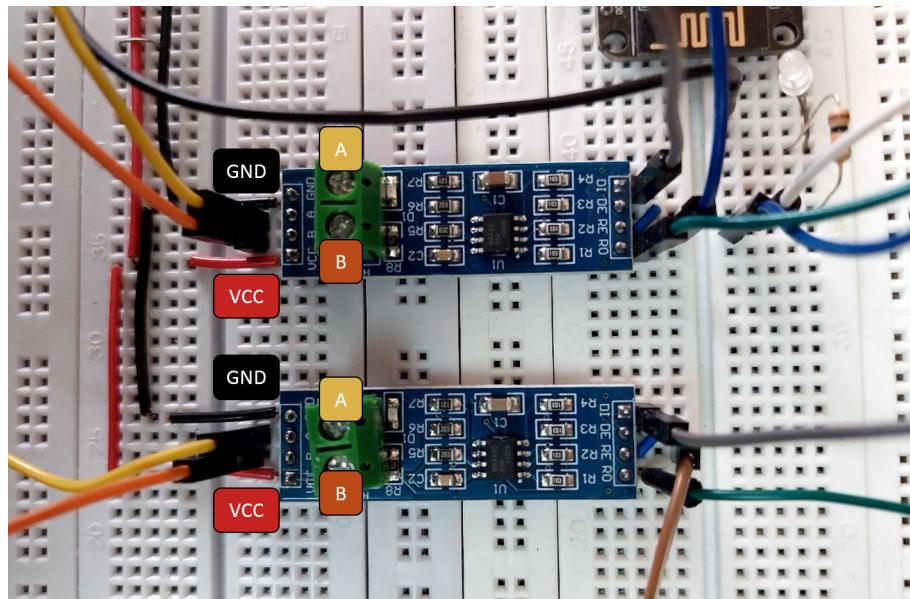


شکل ۶-۳ تصویر اتصالات برد ESP8266 به مازول فرستنده و گیرنده RS485

دو پایه‌ی دیگر مازول RS485 یعنی پایه‌های B و A به ترتیب به پایه‌های B و A دیگر مازول RS485 که به دستگاه بردی M5Stack متصل است، متصل می‌شوند. این اتصالات در شکل ۷-۳ قابل مشاهده هستند.

در ادامه به توضیح اتصالات بین دستگاه بردی M5Stack با مازول RS485 می‌پردازیم. همان‌طور که مشاهده می‌کنید، اتصالات برد ESP32 با مازول RS485 همانند NodeMCU ESP8266 همان‌طور می‌باشد و دیگر نیازی به اتصال پایه‌های VCC و GND نیست. تصویر دقیق‌تر این اتصالات نیز در

<sup>1</sup>GPIO

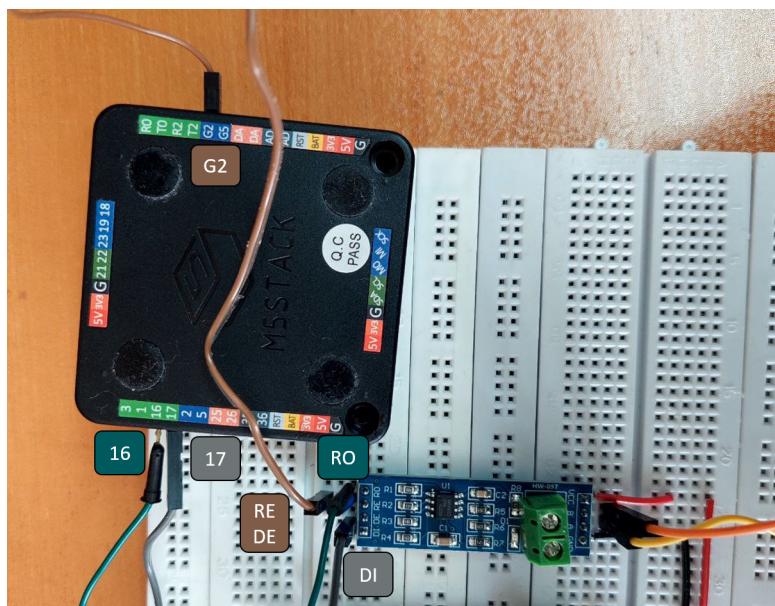


شکل ۷-۳ تصویر اتصالات دو مژول RS485

جدول ۳-۳ جدول اتصالات برد M5Stack به مژول فرستنده و گیرنده RS485

ماژول فرستنده گیرنده RS485	M5Stack
RO	GPIO16 یا پایه‌ی RX
RE و DE	GPIO2
DI	GPIO17 یا پایه‌ی TX

شکل ۸-۳ قابل مشاهده است.



شکل ۸-۳ تصویر اتصالات برد M5Stack به مژول فرستنده و گیرنده RS485

### ۳-۳ ارسال داده‌ها از کارگزار MQTT به پایگاه داده و ذخیره‌ی اطلاعات

هدف از این بخش این است که داده‌هایی که از سمت برد اصلی به سمت کارگزار MQTT ارسال می‌شوند، توسط یک برنامه‌ی پسین<sup>۱</sup> در حال اجرا، در پایگاه داده ذخیره شوند تا امکان دسترسی به داده‌های ذخیره‌شده برای کاربران این سیستم وجود داشته باشد. با آگاهی از هدف ذخیره‌سازی اطلاعات، در این بخش قصد داریم تا نحوه‌ی دریافت داده‌ها از کارگزار و ارسال آن‌ها به سمت پایگاه داده را شرح دهیم.

#### ۱-۳-۳ طراحی و راهاندازی پایگاه داده MySQL

برای اجرا کردن یک برنامه‌ی پسین و اتصال آن به پایگاه داده، پیش از هر چیز لازم است تا یک کارساز به عنوان زیرساخت داشته باشیم. در بخش ۲-۲-۲ گفتیم که در این پروژه برای آماده کردن ماشین مجازی به عنوان زیرساخت جایگزین کارساز جداگانه، از Ubuntu 22.04.2 LTS و WSL2 استفاده کردۀایم. پس از آماده‌سازی زیرساخت، لازم است پایگاه داده را طراحی و سپس راهاندازی کنیم تا در زمان توسعه‌ی برنامه‌ی پسین، امکان اتصال و استفاده‌ی آن را داشته باشیم. هدف از حضور پایگاه داده، ذخیره‌سازی میزان مصرف آب، برق و گاز به طور جداگانه است. به علاوه باید مشخص گردد که میزان مصرف به ازای کدام مشترک است و در چه زمانی این میزان مصرف رخ داده است. با توجه به این موارد، ستون‌ها و محتوای هر یک از آن‌ها در جدول پایگاه داده‌ی مربوط به پروژه، در شکل ۹-۳ قابل مشاهده است.

همانطور که در فصل مفاهیم پایه اشاره شد، پایگاه داده در این پروژه بر روی بستر داکر راهاندازی و مستقر شده است. بدین منظور یک فایل Docker کامپوز نوشته شده است تا تنها با اجرای دستور docker compose up -d در مسیری که این فایل حضور دارد، پایگاه داده‌ی MySQL در پس‌زمینه<sup>۲</sup> راهاندازی شود.

در شکل ۱۰-۳ می‌بینید که از تصویر داکر<sup>۳</sup> از پیش آماده‌ی mysql:8.1.0 برای بالا آوردن سرویس پایگاه داده استفاده شده است. اسم پایگاه داده به همراه نام کاربر و رمز عبور کاربر پایگاه داده، در

<sup>1</sup> Back-end

<sup>2</sup> Background

<sup>3</sup> Docker Image

Action:	ID	BOARD ID	DATA TYPE	CONSUMPTION	TIME
	2	84:0D:8E:3D:53:60	Gas	510.0	Oct. 18, 2023, 6:31 a.m.
	1	84:0D:8E:3D:53:60	Gas	510.0	Oct. 18, 2023, 6:31 a.m.
	3	84:0D:8E:3D:53:60	Water	622.0	Oct. 18, 2023, 6:31 a.m.
	4	84:0D:8E:3D:53:60	Water	622.0	Oct. 18, 2023, 6:31 a.m.
	5	84:0D:8E:3D:53:60	Power	740.0	Oct. 18, 2023, 6:31 a.m.
	6	84:0D:8E:3D:53:60	Power	740.0	Oct. 18, 2023, 6:31 a.m.
	7	84:0D:8E:3D:53:60	Battery	100.0	Oct. 18, 2023, 6:31 a.m.
	8	84:0D:8E:3D:53:60	Battery	100.0	Oct. 18, 2023, 6:31 a.m.
	10	84:0D:8E:3D:53:60	Gas	769.0	Oct. 18, 2023, 6:31 a.m.
	9	84:0D:8E:3D:53:60	Gas	769.0	Oct. 18, 2023, 6:31 a.m.
	11	84:0D:8E:3D:53:60	Water	864.0	Oct. 18, 2023, 6:31 a.m.
	12	84:0D:8E:3D:53:60	Water	864.0	Oct. 18, 2023, 6:31 a.m.
	13	84:0D:8E:3D:53:60	Power	1022.0	Oct. 18, 2023, 6:31 a.m.
	14	84:0D:8E:3D:53:60	Power	1022.0	Oct. 18, 2023, 6:31 a.m.

شکل ۳-۹ جدول پایگاه داده

متغیرهای محیطی<sup>۱</sup> تنظیم شده است. در ادامه مشخص شده که می‌خواهیم این سرویس داکری، بر روی درگاه ۳۳۰۶ اجرا شود. با اضافه کردن volume به فایل داکر کامپوز، از اینکه داده‌های داخل پایگاه داده در اثر پایین آمدن این سرویس از دست بروند، جلوگیری می‌شود.

```

1 version: '3.7'
2
3 services:
4
5   sm_db:
6     image: mysql:8.1.0
7     command: --default-authentication-plugin=mysql_native_password
8     restart: always
9     environment:
10       MYSQL_DATABASE: 'smart_meter'
11       MYSQL_USER: 'smart_meter'
12       MYSQL_PASSWORD: 'B@harKav!ani'
13       MYSQL_ROOT_PASSWORD: 'root@dm!n'
14     ports:
15       - '3306:3306'
16     expose:
17       - '3306'
18     volumes:
19       - sm_volume:/var/lib/mysql
20
21 volumes:
22   sm_volume: {}
23

```

شکل ۳-۱۰ محتوای فایل داکر کامپوز پایگاه داده

<sup>1</sup>Environment Variables

### ۲-۳-۳ طراحی و راهاندازی برنامه‌ی پسین بر روی چارچوب نرم‌افزاری جنگو

همانطور که در بخش‌های قبل اشاره شد، جهت دریافت داده از کارگزار MQTT و ذخیره‌سازی آن در پایگاه داده MySQL که شرح آن در بخش پیشین گذشت، از چارچوب جنگو استفاده می‌کنیم. از آن جایی که این چارچوب براساس الگوی طراحی نرم‌افزاری مدل-نما-کنترلر (MVC)<sup>۱</sup> طراحی شده است نیاز است که تا یک مدل جهت ذخیره‌سازی داده و نیز یک کنترلر جهت مدیریت ورود و خروج داده پیاده‌سازی کنیم. از آنجایی در حال حاضر نیازی به دریافت داده از سرور توسط اشخاص ثالث نیست و در محدوده تعريف این پروژه نمی‌باشد به پیاده‌سازی بخش نما نپرداخته‌ایم.

جهت پیاده‌سازی مدل و دریافت و ذخیره‌ی داده در آن نیاز است که یک اپلیکیشن با نام دلخواه به پروژه اضافه کنیم. در این پروژه، این اپلیکیشن با نام records شناخته می‌شود. این بخش با دستور زیر قابل ایجاد است:

```
Python manage.py startapp records
```

با ایجاد این اپلیکیشن و افزودن آن بخش installed\_apps در فایل settings.py از این پس در پروژه جنگو شناسایی شده و قابل استفاده خواهد بود. در قسمت قبل گفتیم که پایگاه داده‌ی ما به چه ستون‌هایی نیاز دارد. برای طراحی این پایگاه داده در ساختار مدل اپلیکیشن records همانند شکل ۱۱-۳ عمل می‌کنیم.

```
server > records > models.py > ...
You, yesterday | 1 author (You)
1  from django.db import models
2
3  from records.constants import DATA_TYPES
4
5
You, yesterday | 1 author (You)
6  class DataRecord(models.Model):
7      board_id = models.CharField(max_length=64)
8      data_type = models.CharField(max_length=1, choices=DATA_TYPES)
9      consumption = models.FloatField()
10     time = models.DateTimeField()
11
You, yesterday | 1 author (You)
12    class Meta:
13        db_table = "consumption_data_record"
14        ordering = ["time"]
15
16    def __str__(self):
17        return "{} {} consumption from {} board".format(
18            self.consumption, self.data_type, self.board_id
19        )
20
```

شکل ۱۱-۳ مدل طراحی‌شده در اپلیکیشن records

<sup>1</sup>Model–View–Controller

جهت اتصال برنامه‌ی پسین به کارگزار و دریافت داده‌ی ارسالی از سمت برد، لازم است از یک کارخواه MQTT استفاده کنیم و با تنظیم آن، اتصال میان برنامه‌ی پسین و کارگزار را برقرار نماییم. بدین منظور از کتابخانه‌ی paho\_mqtt استفاده کردیم که متعلق به شرکت eclipse می‌باشد. این کتابخانه به علت پشتیبانی بلند مدت و قوی و نیز سندهای دقیق یکی از پرکاربردترین و قابل اتكاترین کتابخانه‌های موجود در پایتون جهت استفاده می‌باشد [۲۱].

در هر پروژه جنگو تمام تنظیمات ابزارهای مورد استفاده در پروژه، در فایل settings.py که در پوشه‌ی اصلی پروژه جای گرفته، قرار می‌گیرد. به این منظور آدرس، نام کاربری، گذر واژه و تمام اطلاعات مرتبط با چگونگی اتصال به کارگزار MQTT را در این بخش قرار می‌دهیم. تنظیم keep\_alive باعث می‌شود تا در صورتی که در طول مدت تعیین شده هیچ داده‌ای بین برنامه و کارگزار رخ نداد ارتباط قطع شود ولی همواره از آخرین دریافت یا ارسال داده به همین اندازه اتصال را برقرار نگه دارد. در شکل ؟؟ می‌توانید طریقه‌ی مقداردهی به این تنظیمات را ببینید.

```

147
148     # MQTT Configurations
149     MQTT_SERVER = "test.mosquitto.org"
150     MQTT_PORT = 1884
151     MQTT_KEEPALIVE = 60
152     MQTT_USER = "rw"
153     MQTT_PASSWORD = "readwrite"
154

```

شکل ۱۲-۳ تنظیمات اتصال به کارگزار MQTT در چارچوب جنگو

با داشتن تنظیمات، برنامه‌ی پسین اطلاعات بر روی موضوعات مربوط به آب، برق و گاز را که در ادامه نیز آورده شده است، پس از طی کردن مرحله‌ی احراز هویت و اتصال به کارگزار MQTT دریافت خواهد کرد.

AUTSmartMeteringSystem/gas/<Board ID>/consumption •

AUTSmartMeteringSystem/water/<Board ID>/consumption •

AUTSmartMeteringSystem/power/<Board ID>/consumption •

حال برنامه می‌تواند با کمک این موضوعات و داده‌هایی که بر روی آن‌ها دریافت کرده، اطلاعات را پردازش و جداسازی کند و سپس آن‌ها را در پایگاه داده از نوع MySQL که از قبل جدول آن نیز طراحی شده، ذخیره کند.

## فصل چهارم

### چالش‌ها و محدودیت‌های انجام پروژه

در این فصل، به توضیح برخی از چالش‌هایی که در مسیر انجام پروژه به آن‌ها برخور迪م، خواهیم پرداخت. به طور کلی می‌توان چالش‌های اصلی این پروژه را در چهار دسته‌ی فرایندی، معماری، سخت‌افزاری و نرم‌افزاری دسته‌بندی کرد.

### ۱-۴ چالش‌های فرایندی

در این بخش به بررسی انواع چالش‌هایی که در ارتباط با سایر بخش‌های سامانه‌ی هوشمند اندازه‌گیری مصرف در ساختمان است، می‌پردازیم. از آنجا که درواره‌ی مرکزی، وظیفه‌ی برقراری ارتباط با بخش‌های نرم‌افزاری و سخت‌افزاری را دارد، برای ارتباط با هر یک از طرفین چالش‌های متفاوتی را دارد که در ادامه به توضیح اصلی‌ترین آن‌ها خواهیم پرداخت.

#### ۱-۱-۴ تعیین فرایند برای چگونگی ارتباط با سامانه‌ی نرم‌افزاری

در این پروژه، یکی از چالش‌های اساسی، برقراری ارتباط بین بخش‌های سخت‌افزاری و نرم‌افزاری بود. این موضوع نیازمند همکاری مداوم با پروژه‌های توسعه‌ی نرم‌افزار و سخت‌افزار بود تا اطمینان حاصل شود که خدمات ما می‌توانند به طور هماهنگ با هم کار کنند.

برای رسیدن به هدف برقراری ارتباط مؤثر با بخش نرم‌افزاری، حفظ ارتباط دائمی با توسعه‌دهنده نرم‌افزار، مشارکت فعالانه در طراحی موضوعات MQTT و اصلاح مشترک پروتکل‌های ارتباطی ضروری بود. به علاوه، بررسی‌های منظم و اعتبارسنجی اتصال بین درواره‌ی ارتباطی مرکزی و بخش نرم‌افزاری برای تضمین صحت و قابلیت اطمینان داده‌ها حیاتی بود.

#### ۲-۱-۴ انتخاب پروتکل برای ارتباط با کنتور سخت‌افزاری

انتخاب پروتکل ارتباطی مناسب بین سخت‌افزار کنتور و میان‌افزار یکی دیگر از چالش‌های مهمی بود که در طی انجام پروژه به آن برخور迪م. وجود استانداردهای متعدد برای برقراری ارتباط فیزیکی بین دستگاه‌ها، اهمیت تصمیم‌گیری آگاهانه را بالاتر می‌برد.

استاندارد ارتباطی فیزیکی RS485 یک استاندارد ثبت‌شده و قابل اعتماد است که از استحکام انتقال داده در شرایط مختلف اطمینان می‌دهد. این پروتکل قادر است تداخل‌های ناشی از وجود اختلالات الکتریکی را که در محیط‌های مختلف به ویژه محیط‌های صنعتی رایج است، به طور مؤثر کاهش دهد.

به علاوه، این استاندارد، به دلیل استفاده از کابل‌های ارتباطی ساده و ارزان، هزینه‌ی نصب و پیاده‌سازی را به طور چشم‌گیری کاهش می‌دهد. این ویژگی برای پروژه‌هایی که نیاز به اتصالات پیچیده و همچنین انتقال داده در فوائل بزرگ دارند، بسیار ارزشمند است.

پس از ارزیابی‌های انجام شده، استاندارد RS485 به دلایل مختلفی به عنوان پروتکل ارتباطی بین سخت‌افزار شبیه‌ساز کنترلر و میان‌افزار انتخاب شد.

### ۲-۴ چالش‌های معماری

#### ۱-۲-۴ انتخاب موضوعات مناسب بر روی پروتکل MQTT

در این پروژه، زمانی که داده‌های مربوط به کنترلهای مختلف (آب، برق و گاز) باید به بخش نرم‌افزار منتقل شوند، انتخاب موضوعات مناسب MQTT بسیار مهم است. یکی از ملاحظات مهم در زمان کار با پروتکل MQTT طراحی دقیق موضوعات است که می‌تواند اثربخشی کار با این پروتکل را افزایش دهد؛ به ویژه در حالتهایی همانند این پروژه که افراد و سازمان‌های مختلفی از جمله مصرف‌کنندگان در سطح شهر و کشور و سازمان‌های مختلف تامین‌کننده‌ی انرژی و منابع در آن نقش ایفا می‌کنند و داده‌ها باید برای اهداف و ذی‌نفعان مختلف تفکیک شوند.

چالش‌ها در این زمینه چند وجهی هستند:

۱- جداسازی داده‌ها: برای اطمینان از اینکه هر نوع داده‌ی مصرفی (آب، برق و گاز) می‌تواند به طور جداگانه منتقل شود، طراحی یک سلسله مراتب موضوعی واضح و سازمان‌یافته ضروری است. این امر از همپوشانی<sup>۱</sup> داده‌ها جلوگیری می‌کند و به هر سازمان یا بخش اجازه می‌دهد تا بدون سردرگمی به داده‌های مربوط به خود دسترسی داشته باشد.

۲- امنیت و کنترل دسترسی<sup>۲</sup>: موضوعات MQTT نقش مهمی در اجرای اقدامات امنیتی و کنترل دسترسی ایفا می‌کنند. برای مثال، می‌خواهید مطمئن شوید که داده‌های حساس، مانند میزان مصرف برق مشتریان، تنها برای کارکنان یا سیستم‌های مجاز قابل دسترسی است. در این صورت

<sup>1</sup>Overlap

<sup>2</sup>Access Control

تعیین چگونگی ساختار موضوعات، در حالی که جریان داده به طور کارآمد حفظ شود و اقدامات امنیتی نیز در نظر گرفته شود، یک چالش مهم است.

۳- سازگاری<sup>۱</sup>: اطمینان از یکنواختی در قراردادهای نام‌گذاری موضوعات برای نگهداری طولانی‌مدت سیستم حیاتی است. توجه به این موضوع می‌تواند فرایند نام‌گذاری مداوم، اضافه کردن دستگاه‌های جدید یا بهروزرسانی دستگاه‌های موجود را بدون ایجاد سردرگمی و آسان کند.

#### ۲-۲-۴ امنیت داده‌های ارسالی

بدیهی است که اطمینان از امنیت داده‌های در حال انتقال بسیار مهم است. اطمینان از امنیت داده‌های ارسالی از اهداف اصلی این پروژه نبوده است، اما با این وجود اقداماتی برای محترمانه بودن و اعتبار داده‌ها انجام داده شده که در ادامه توضیح جامعی از اقدامات امنیتی انجام شده، آورده شده است:

فرایند احراز هویت<sup>۲</sup>: برای ایمن‌سازی داده‌های ارسالی از کنتورها به سیستم نرم‌افزاری مرکزی، فرایند احراز هویت اجرا می‌شود. این موضوع تضمین می‌کند که فقط دستگاه‌ها یا کاربران مجاز می‌توانند با کارگزار MQTT تعامل داشته باشند و از دسترسی غیرمجاز به داده‌ها جلوگیری می‌کند. احراز هویت معمولاً شامل استفاده از اعتبارنامه‌های منحصر به فرد<sup>۳</sup>، مانند نام کاربری و رمز عبور یا سایر اشکال احراز هویت است. در این پروژه از نام کاربری و رمز عبور به منظور دستیابی به اطلاعات استفاده می‌شود.

ارتباط امن: فرایند انتقال داده با رمزنگاری ارتباط بین درواوهای مرکزی و کارگزار MQTT ایمن می‌شود. پروتکل‌های رمزنگاری، مانند TLS/SSL، معمولاً برای ایجاد کانال‌های ارتباطی امن استفاده می‌شوند. این رمزنگاری تضمین می‌کند که حتی اگر مهاجم<sup>۴</sup>، بسته‌های داده در حال انتقال را رهگیری کند، نمی‌تواند محتويات را بدون کلیدهای رمزگشایی مناسب رمزگشایی کند.

کنترل دسترسی: مکانیسم‌های کنترل دسترسی برای محدود کردن افرادی که می‌توانند داده‌ها را بخوانند یا بنویسند در موضوعات خاص MQTT ایجاد شده است. به عنوان مثال، با استفاده از نرم‌افزار MQTT Explorer، تنها با وارد کردن اطلاعات نام کاربری و رمز عبور می‌توان به کارگزار MQTT متصل شد و به اطلاعات موجود بر روی موضوعات این سامانه دست پیدا کرد. در شکل ۱-۴ تصویری از اطلاعات

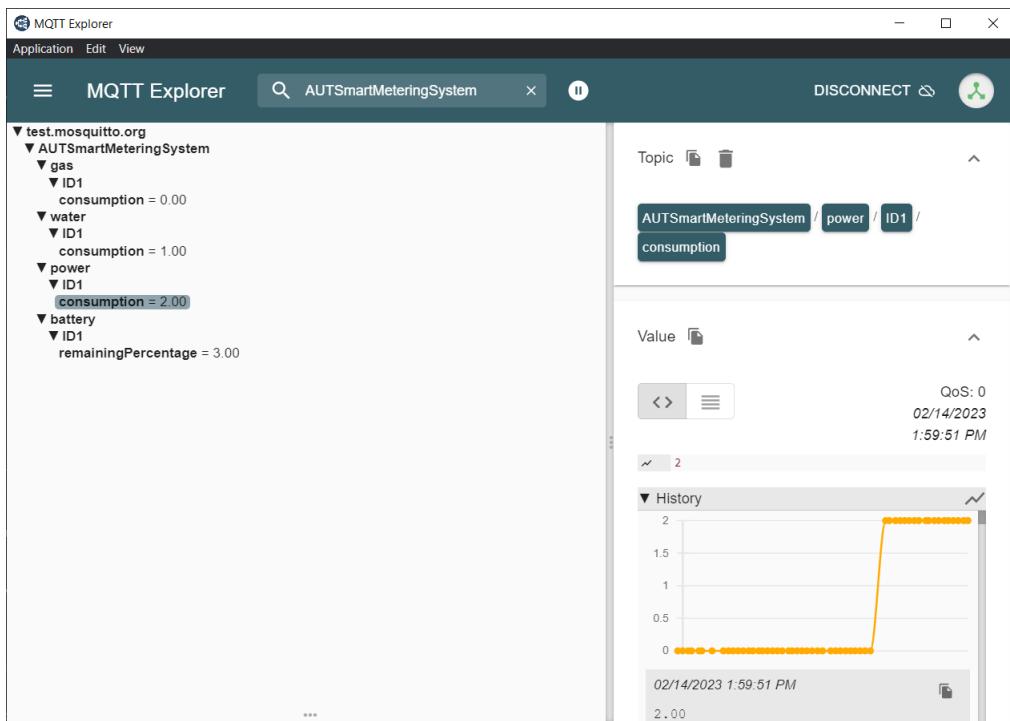
<sup>1</sup>Consistency

<sup>2</sup>Authentication

<sup>3</sup>Unique Credentials

<sup>4</sup>Adversary

مختلف بر روی موضوعات پس از طی کردن فرایند احراز هویت و اتصال به کارگزار را مشاهده می‌کنید.



شکل ۱-۴ تصویری از محیط نرمافزار MQTT Explorer

این کنترل دسترسی تضمین می‌کند که فقط کاربرانی که اعتبارنامه صحیحی دارند می‌توانند در موضوعات MQTT مشترک شوند یا داده‌ها را منتشر کنند.

### ۳-۴ چالش‌های سخت‌افزاری

در این بخش به برخی از چالش‌هایی که در بخش سخت‌افزار بردها و اتصالات بین آن‌ها بود، می‌پردازیم.

#### ۱-۳-۴ تولید ورودی‌های آزمون

در طول پروژه، یکی از چالش‌های مهم مرتبط با سخت‌افزار، ایجاد یک محیط مناسب برای آزمایش و اعتبارسنجی عملکرد سیستم اندازه‌گیری بود. در اینجا توضیح مفصلی از این چالش آورده شده است. آزمون<sup>۱</sup>، مرحله‌ای حیاتی در توسعه‌ی هر سیستمی است؛ به ویژه در پروژه‌هایی که شامل یکپارچه‌سازی سخت‌افزار و نرم‌افزار است. وجود آزمون تضمین می‌کند که تمامی اجزا به طور هماهنگ و صحیح کار

<sup>1</sup>Test

می‌کنند و داده‌ها به طور دقیق جمع‌آوری، انتقال و پردازش می‌شوند.

برای غلبه بر کمبود داده‌های دنیای واقعی و شبیه‌سازی رفتار سیستم کامل، باید داده‌های آزمایشی را طراحی و ایجاد می‌کردیم. این طراحی، شامل تولید داده‌های شبیه‌سازی شده است که خوانش‌ها و رفتارهای مورد انتظار از کنتورهای واقعی را تقلید می‌کند. داده‌های آزمون، به ما این امکان را می‌دهد که فرایند دریافت ورودی توسط برد اصلی را آزمایش کنیم و اطمینان حاصل کنیم که داده‌ها را به درستی از اندازه‌گیرهای سخت‌افزار دریافت می‌کند.

### ۲-۳-۴ برقراری اتصالات مناسب

اطمینان از اتصالات صحیح و دقیق یکی از جنبه‌های اساسی هر پروژه‌ی سخت‌افزاری است. RS485 یک استاندارد ارتباطی پرکاربرد است که به دلیل استحکام و تطبیق‌پذیری آن شناخته شده است. با این حال، داشتن درک کامل از مدل اتصال RS485، که شامل جنبه‌هایی مانند سیم‌کشی، سیگنال‌دهی و اتمام پیام است، بسیار مهم است. همانطور که در فصل دوم نیز به توضیح کامل استاندارد RS485 پرداخته شد، می‌دانیم که این ارتباط معمولاً از دو سیم برای ارتباط استفاده می‌کند: یکی برای انتقال داده (TX) و دیگری برای دریافت داده (RX). همچنین به ملاحظات اضافی مانند اتصال زمین برای حفظ یکپارچگی سیگنال در فواصل طولانی‌تر نیاز است.

لازم به ذکر است که همه‌ی بردات توسعه یا میکروکنترلرهای ارتباط RS485 را به طور مستقیم پشتیبانی نمی‌کنند. بنابراین، بررسی سازگاری برد اصلی M5Stack و همچنین برد شبیه‌ساز ESP8266 با ارتباطات RS485 ضروری است. در حقیقت، اجزای سخت‌افزاری اضافی استفاده شده در پروژه مانند ماژول فرستنده گیرنده RS485 یا ماژول‌های مبدل برای فعال کردن ارتباط RS485 روی برد اصلی لازم است.

### ۳-۳-۴ پیکربندی ارتباط سریال

پیکربندی صحیح ارتباط سریال بسیار مهم است. از آنجا که ما از پروتکل Modbus در این پروژه استفاده کرده‌ایم، پیکربندی ارتباط شامل تنظیم عامل‌هایی مانند نرخ باد<sup>۱</sup> و ثبات‌های<sup>۲</sup> داده برای مطابقت

<sup>1</sup>Baud rate

<sup>2</sup>Register

با الزامات ارتباطی شبکه RS485 است. عدم تطابق تنظیمات می‌تواند منجر به خطاهای ارتباطی و از دسترفتن داده‌ها شود.

هم چنین ارتباطات RS485 اغلب شامل چندین گره یا دستگاهی هستند که در یک گذرگاه با هم ارتباط برقرار می‌کنند. بنابراین قبل از شروع کار با توابع کتابخانه‌ی Modbus، اطمینان از اینکه هر گره دارای یک آدرس یا شناسه منحصر به فرد است برای جلوگیری از برخورد داده‌ها و رفع مشکلات مسیریابی ضروری است.

#### ۴-۳-۴ نحوه بررسی ارسال داده بین دو برد

پس از ایجاد اتصالات فیزیکی، آزمایش و اعتبارسنجی دقیق ضروری است. این امر شامل ارسال و دریافت داده‌های آزمایشی بین برد اصلی و برد شبیه‌ساز است؛ به منظور آن که بتوان درستی اتصالات بین بردها و صحت داده‌های ارسالی بین آن دو را مورد آزمایش قرار داد.

در مراحل اولیه پروژه، ابهاماتی در تعیین دقیق اینکه آیا مشکلات انتقال داده از ESP8266 (برد ارسال داده) یا M5Stack (برد اصلی) نشأت می‌گیرد، وجود داشت. همچنین تشخیص اینکه آیا مشکل مربوط به سخت‌افزار است یا کد، عیوب‌یابی<sup>۱</sup> را پیچیده‌تر می‌کند. برای رفع این چالش، یک تراشه‌ی سخت‌افزاری مبدل USB به RS485 را در راهاندازی اتصالات استفاده کردیم تا امکان تست انتقال داده و اشکال‌زدایی کارآمد را انجام دهد.

ابتدا، اشکال‌زدایی<sup>۲</sup> و بررسی عملکرد ارسال داده در برد ESP8266 انجام گرفت. برای انجام این کار، یک سر اتصال RS485 را به برد ESP8266 و سر دیگر را به کمک مبدل USB-RS485 به رایانه‌ی شخصی متصل کردیم. با این تنظیمات، برد ESP8266 به عنوان برد ارباب عمل می‌کند و انتقال داده‌ها را آغاز می‌کند. برای دریافت و تفسیر این داده‌ها در رایانه، نرم افزار Modbus Slave برای ارسال داده‌ها به برد M5Stack استفاده شد. این مرحله امکان بررسی و اشکال‌زدایی این

گرفت.[۲۲].

در مرحله‌ی دوم، بین M5Stack و رایانه‌ی شخصی ارتباط برقرار شد. در این حالت، رایانه به عنوان ارباب عمل کرده که این بار نرم‌افزار Modbus Slave دیگر قابل اجرا نبود. در عوض، از نرم‌افزار Poll برای ارسال داده‌ها به برد M5Stack استفاده شد. این مرحله امکان بررسی و اشکال‌زدایی این

<sup>1</sup>Troubleshooting

<sup>2</sup>Debug

موضوع است که آیا M5Stack با موفقیت داده‌های ارسال شده از رایانه را دریافت کرده است یا خیر [۲۳]. در نهایت، پس از اطمینان از ارسال داده از سمت برد ESP8266 و دریافت داده در سمت M5Stack چالش باقی‌مانده شامل اطمینان از پیکربندی صحیح اتصالات فیزیکی و سیم‌کشی بین دو برد ESP8266 و M5Stack بود که پیش‌تر توضیح داده شد.

### ۴-۴ چالش‌های نرم‌افزاری

در این بخش به برخی از چالش‌هایی که در بخش برنامه‌نویسی بردها به آن‌ها برخوردیم، می‌پردازیم.

#### ۱-۴-۱ نحوه‌ی بررسی ارسال داده بین برد اصلی و کارگزار MQTT

یکی از چالش‌های مهم نرم‌افزاری در پروژه، اطمینان از انتقال یکپارچه داده‌ها بین برد اصلی و کارگزار MQTT است. برای حل این چالش، به طور مؤثر از نرم‌افزار MQTT Explorer استفاده شده است. این ابزار این امکان را می‌دهد که به داده‌ها در هنگام ارسال به کارگزار و نحوه‌ی نمایش آن‌ها نظارت داشته باشیم. با استفاده از MQTT Explorer، می‌توان تأیید کرد که داده‌ها به درستی از برد اصلی به کارگزار MQTT منتقل شده‌اند، که نظارت کارآمد داده‌ها و عیوب‌یابی را تسهیل می‌کند.

#### ۲-۴-۲ لزوم آشنایی کامل با سخت‌افزار برد در هنگام توسعه‌ی برنامه

توسعه‌ی نرم‌افزار مناسب برای هر بردی نیاز به درک جامعی از سخت‌افزار آن برد دارد. این آشنایی برای نوشتن کدی که می‌تواند به طور مؤثر با اجزای سخت‌افزاری و حسگرهای درگیر در جمع‌آوری و انتقال داده‌ها ارتباط برقرار کند، بسیار مهم است.

به عنوان مثال درگاه سریال برد ESP8266 زمانی که این برد به رایانه متصل است و از رایانه تغذیه می‌کند، به دلیل وجود این ارتباط، اشغال خواهد شد. بنابراین در زمانی که ما نیاز داریم تا ارتباط RS485 را روی این برد داشته باشیم، نمی‌توانیم از پایه‌های سریال خود برد استفاده کنیم. بنابراین از دو پایه‌ی ورودی-خروجی عمومی بر روی این برد استفاده کرده‌ایم و در عوض از کتابخانه‌ی SoftwareSerial در کد استفاده کرده‌ایم تا بتوانیم هم‌چنان ارتباط سریال را در این برد داشته باشیم.

## فصل پنجم

### جمع‌بندی و پیشنهادها

## ۱-۵ جمع‌بندی

در این پژوهه، ارتباط بین سخت‌افزار کنتورهای آب، برق و گاز و نرم‌افزار تحت وب برای سامانه‌ی اندازه‌گیری هوشمند با پیاده‌سازی درواهی مرکزی ارتباط، برقرار شده است. ابتدا با نیازمندی‌ها آشنا شدیم و سپس به شناسایی ابزار و فناوری‌های مورد نیاز برای پیشبرد این پژوهه پرداختیم.

پس از آشنایی با مفاهیم اولیه، روند پیاده‌سازی پژوهه در سه فاز مختلف انجام گرفت: در قسمت اول، اتصال بین برد اصلی با کارگزار MQTT برقرار شد. سپس در قسمت دوم، برد شبیه‌سازی از داده‌های کنتورها به پژوهه اضافه شد تا بتواند ورودی‌های دنیای بیرونی را برای برد اصلی شبیه‌سازی کند. در این قسمت لازم بود که برقراری ارتباط بین دو برد با یک استاندارد و پروتکل در لایه‌ی فیزیکی انجام گیرد و برای این منظور، استاندارد RS485 انتخاب شد. در قسمت سوم نیز داده‌هایی که از سمت کنتور دریافت شده و به سمت کارگزار MQTT ارسال می‌شوند، از سمت کارگزار دریافت و در یک پایگاه داده از نوع MySQL ذخیره می‌شوند.

بخش کار با بردها در این پژوهه به صورت متن‌باز در وبگاه گیتاب<sup>۱</sup> قابل دسترسی است. همچنین، مستندات لازم برای شیوه کار با آن نیز در این وبگاه به تفصیل شرح داده شده است. برخی از مسیرها و بخش‌های مهم در این وبگاه، برای دسترسی سریع‌تر به منابع در ادامه توضیح داده شده است:

-۱ مسیر Central\_gateway\_for\_Smart\_metering\_system > src > connect\_to\_mqtt\_broker

برنامه و مستندات مربوط به پیاده‌سازی برد اصلی در این بخش قابل دسترسی هستند.

-۲ مسیر Central\_gateway\_for\_Smart\_metering\_system > src > send\_simulated\_data

برنامه و مستندات مربوط به پیاده‌سازی برد شبیه‌ساز در این بخش قابل دسترسی هستند.

-۳ مسیر Central\_gateway\_for\_Smart\_metering\_system > example\_code

علاوه بر بخش‌های اصلی پژوهه، برخی از پیاده‌سازی‌های مهم پژوهه به صورت مثال این مسیر نیز آورده شده است.

---

<sup>1</sup>[https://github.com/Baharkaviani/Central\\_gateway\\_for\\_Smart\\_metering\\_system](https://github.com/Baharkaviani/Central_gateway_for_Smart_metering_system)

:Central\_gateway\_for\_Smart\_metering\_system > Docs - ۴ مسیر

برای اطلاع از منابع یا آدرس‌های اینترنتی مفیدی که در حین پروژه استفاده شده‌اند می‌توانید به این مسیر مراجعه کنید. در همین مسیر با مطالعه‌ی بخش m5stack/m5stack\_quickstart آشنایی کنید. در این پروژه، یعنی برد M5Stack آشنا شوید و سپس بتوانید برنامه‌ی مربوط به پروژه را بر روی برد خود اجرا و آزمایش کنید.

بخش دیگر پروژه که شامل فایل‌های مربوط به پایگاه داده و برنامه‌ی پسین می‌باشد نیز به صورت متن‌باز در وبگاه گیتاب<sup>۱</sup> قرار داده شده است.

## ۲-۵ پیشنهادها و کارهای آینده

این پروژه تنها بخشی از یک هدف بسیار بزرگ‌تر، یعنی هدف افزایش کیفیت خدمات ضروری مانند آب، برق و گاز است که به خودی خود رویکری دقیق برای رسیدگی به چالش‌های کلیدی مرتبط با اندازه‌گیری این‌گونه مصارف و پردازش هوشمند داده‌های جمع‌آوری‌شده است. سیستم‌های اندازه‌گیری سنتی و غیرهوشمند منجر به ناکارآمدی و افزایش هزینه‌ها می‌شوند. پیاده‌سازی سیستم‌های اندازه‌گیری هوشمند اولین گام در راه حرکت به سمت شبکه‌ی هوشمند و یکی از عنصرهای کلیدی زیرساخت اندازه‌گیری پیشرفت‌های می‌باشد.

بنابراین این پروژه تنها بخش کوچکی از یک پروژه‌ی بسیار گسترده است که کارهای بسیار زیادی می‌تواند در راستای بهبود آن و صنعتی‌تر شدن آن برداشته شود. از مهم‌ترین اقداماتی که برای بهبود این پروژه می‌توان انجام داد، مقیاس‌پذیرتر کردن آن است. همانطور که پروژه پیشرفت می‌کند و اندازه‌گیری مصارف یکپارچه می‌شوند، تعداد برد‌ها نیز افزایش می‌یابد و در این بین باید برای تعداد زیادی برد در سطح شهر و کشور، شناسه‌ی منحصر به فرد تخصیص داده شود. بنابراین باید به دنبال راه حل‌هایی برای مدیریت هزینه و افزایش مقیاس‌پذیری در تعداد دستگاه‌ها و میزان داده‌ها باشیم. هم‌چنین علاوه بر مقیاس‌پذیرتر کردن برنامه می‌توان فرآیند تولید داده‌های آزمایشی را هم مقیاس‌پذیر کرد تا آزمایش تعداد رو به رشد مصارف، در کنار افزایش تعداد برد‌ها در سطح شهر و کشور را بدون افزایش قابل توجهی در حجم کار لازم برای راه‌اندازی محیط آزمایش انجام دهد.

<sup>1</sup>[https://github.com/Baharkaviani/Central\\_gateway\\_backend](https://github.com/Baharkaviani/Central_gateway_backend)

از دیگر کارهای پیشنهادی این است که برای ساده‌سازی محیط آزمون، تا حد امکان فرآیند تولید داده‌های آزمون را خودکار کنیم. ابزارها یا برنامه‌های خودکارسازی می‌توانند به تولید حجم زیادی از داده‌های آزمایشی به طور کارآمد و مداوم کمک کنند.

نکته‌ی مهم دیگر در مورد پروژه این است که ماهیت داده‌های مورد استفاده، شامل ثبت مصرف منابع در طول زمان است. پایگاه داده‌های رابطه‌ای سنتی مانند MySQL ممکن است انتخاب بهینه‌ای در برخورد با داده‌های سری زمانی که حجم آن به طور مداوم افزایش می‌یابد، نباشد. این به این دلیل است که پایگاه داده‌های رابطه‌ای می‌توانند در هنگام مدیریت حجم زیادی از داده‌های سری زمانی ناکارآمد و نیازمند منابع زیادی باشند.

برای رفع این چالش، می‌توان از InfluxDB که یک پایگاه داده سری زمانی با کارایی بالا و متن‌باز است استفاده کرد. InfluxDB برای ذخیره سازی و پرس‌وجوی کارآمد داده‌های سری زمانی ساخته شده است. InfluxDB مقیاس‌پذیری افقی را فراهم می‌کند، به این معنی که با افزایش حجم داده‌ها، می‌توانید به راحتی زیرساخت پایگاه داده‌ی خود را مقیاس‌بندی کنید. این موضوع تضمین می‌کند که سیستم شما می‌تواند تعداد فزاینده‌ی اندازه‌گیری و داده‌ها را بدون به خطر اندختن عملکرد کنترل کند.

برای افزایش امنیت داده‌ها لازم است که تنها به مکانیزم احراز هویت اتکا نکنیم و راهکارهای دیگری را هم برای محافظت از داده‌ها در این میان داشته باشیم. امنیت داده‌ها مسئله‌ای حیاتی است. می‌توان روش‌های پیشرفته‌تری برای امنیت داده‌ها پیاده‌سازی کرد، از جمله رمزنگاری داده‌ها و احراز هویت دو مرحله‌ای برای دسترسی به داده‌ها.

با ادغام هوش مصنوعی و اینترنت اشیا، می‌توان به صورت هوشمندانه‌تری داده‌ها را تجزیه و تحلیل کرد. مثلاً می‌توان پیش‌بینی کرد که در آینده کدام مناطق نیاز به بیشترین تأمین انرژی دارند. یا بر اساس الگوهای مصرف گذشته، می‌توان پیش‌بینی کرد که در کدام ساعت‌ها نیاز به انرژی بیشتری داریم و در نتیجه با کمک این اطلاعات و به کارگیری آن‌ها از انرژی بیشتری استفاده کرد.

# مراجع

- [1] Kashyap, Monika, Sharma, Vidushi, and Gupta, Neeti. Taking mqtt and nodemcu to IoT: communication in internet of things. *Procedia Computer Science*, 132:1611–1618, 2018.
- [2] Wortmann, Felix and Flüchter, Kristina. Internet of things: technology and value added. *Business & Information Systems Engineering*, 57:221–224, 2015.
- [3] Farhan, Laith, Shukur, Sinan T, Alissa, Ali E, Alrweg, Mohmad, Raza, Umar, and Kharel, Rupak. A survey on the challenges and opportunities of the internet of things (IoT). in 2017 Eleventh International Conference on Sensing Technology (ICST), pp. 1–5. IEEE, 2017.
- [4] Lee, Michael, Aslam, Omar, Foster, Ben, Kathan, David, Kwok, Jordan, Medearis, Lisa, Palmer, Ray, Sporborg, Pamela, and Tita, Michael. Assessment of demand response and advanced metering. Federal Energy Regulatory Commission, Tech. Rep, 2013.
- [5] Al-Ali, Abdul-Rahman, Zualkernan, Imran A, Rashid, Mohammed, Gupta, Ragini, and AliKarar, Mazin. A smart home energy management system using IoT and big data analytics approach. *IEEE Transactions on Consumer Electronics*, 63(4):426–434, 2017.
- [6] educba. What is arduino? Available from: <https://www.educba.com/what-is-arduino/>, June 2023.

- [7] raspberrypi. raspberrypi. Available from: <https://www.raspberrypi.com/for-industry/>.
- [8] m5stack. m5stack. Available from: <https://m5stack.com/>.
- [9] circuitstate. What is rs-485 how to use max485 with arduino for reliable long-distance serial communication. Available from: <https://www.circuitstate.com/tutorials/what-is-rs-485-how-to-use-max485-with-arduino-for-reliable-long-distance-serial-communication/>. April 2023.
- [10] Hunkeler, Urs, Truong, Hong Linh, and Stanford-Clark, Andy. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. in 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), pp. 791–798. IEEE, 2008.
- [11] HiveMQ. Mqtt essentials. Available from: <https://www.hivemq.com/mqtt-essentials/>, August 2023.
- [12] Mishra, Biswajeeban. Performance evaluation of mqtt broker servers. in International Conference on Computational Science and Its Applications, pp. 599–609. Springer, 2018.
- [13] Koziolek, Heiko, Grüner, Sten, and Rückert, Julius. A comparison of mqtt brokers for distributed iot edge computing. in Software Architecture: 14th European Conference, ECSA 2020, L’Aquila, Italy, September 14–18, 2020, Proceedings 14, pp. 352–368. Springer, 2020.
- [14] mosquitto. mosquitto. Available from: <https://mosquitto.org/>.
- [15] Canonical-Ltd. Ubuntu on wsl. Available from: <https://ubuntu.com/wsl>.

- [16] Docker-Inc. Docker overview. Available from: <https://docs.docker.com/get-started/overview/>.
- [17] Django-Software-Foundation. Meet django. Available from: <https://www.djangoproject.com/>.
- [18] modbus. modbus. Available from: <https://modbus.org/>.
- [19] Fovino, Igor Nai, Carcano, Andrea, Maserà, Marcelo, and Trombetta, Alberto. Design and implementation of a secure modbus protocol. in Critical Infrastructure Protection III: Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Hanover, New Hampshire, USA, March 23-25, 2009, Revised Selected Papers 3, pp. 83–96. Springer, 2009.
- [20] Emelianov, Alexander. modbus-esp8266 library. Available from: <https://github.com/emelianov/modbus-esp8266>.
- [21] pypi. paho-mqtt library. Available from: <https://pypi.org/project/paho-mqtt/>.
- [22] modbustools. Modbus slave. Available from: [https://www.modbustools.com/modbus\\_slave.html](https://www.modbustools.com/modbus_slave.html).
- [23] modbustools. Modbus poll. Available from: [https://www.modbustools.com/modbus\\_poll.html](https://www.modbustools.com/modbus_poll.html).

# واژه‌نامه‌ی فارسی به انگلیسی

Subscribe .....	اشتراک .....	ا
Debug .....	اشکال‌زدایی .....	آدرس MAC
Unique .....	اعتبارنامه‌های منحصر‌به‌فرد .....	Address (MAC Address)
Credentials .....	Credentials .....	آردوینو .....
Publish .....	انتشار .....	آزمون .....
Internet of Things (IoT) .....	اینترنت اشیا .....	آی‌پی .....
ب		اتحادیه صنایع الکترونیک .....
Power .....	باتری .....	Industries Alliance
Slave .....	برده .....	احراز هویت .....
پ		اختال .....
Pin .....	پایه .....	common-mode noise
GPIO .....	پایه‌های ورودی و خروجی عمومی .....	ارباب .....
Database .....	پایگاه داده .....	استاندارد توصیه‌شده .....
Modular .....	پیمانه‌ای .....	Standard .....
ت		استحکام .....

Raspberry Pi .....	رزبری پای .....	I	Impedance matching .....	تطبیق امپدانس .....
س	.....	S	Chip .....	تراشه .....
Consistency .....	سازگاری .....	C	single-ended .....	تک‌پایانی .....
Customization .....	سفارشی‌سازی .....	C	.....	ث
Smart Metering .....	سیستم اندازه‌گیری هوشمند .....	S	Register .....	ثبت .....
System .....	.....	S	.....	H
system-on-chip (SoC) .....	سیستم روی تراشه .....	S	RAM .....	حافظه .....
Differential signalling .....	سیگنال‌دهی تفاضلی .....	D	.....	X
ش	.....	S	.....	.....
Smart Grid .....	شبکه‌ی هوشمند .....	S	Smart Home .....	خانه‌ی هوشمند .....
Simulated .....	شبیه‌سازی‌شده .....	S	Gateway .....	درواره .....
ص	.....	S	.....	D
Lcd .....	صفحه نمایش .....	L	Central .....	درواره‌ی ارتباطی مرکزی .....
ط	.....	L	Communication Gateway .....	.....
Schematic .....	طرح‌واره .....	S	Port .....	درگاه .....
ع	.....	S	Ethernet Port .....	درگاه اترنت .....
Troubleshooting .....	عیب‌یابی .....	T	.....	R
غ	.....	T	Driver .....	راهانداز .....
non-inverting .....	غیرمعکوس .....	N	Event .....	رویداد .....
ق	.....	C	Cable .....	رسیمان .....

Overlap .....	همپوشانی .....	قابل انباشته .....
9		قابل حمل .....
Remote Terminal .....	واحد ترمینال از راه دور ..	م
	Unit (RTU)	
GPU .....	واحدهای پردازش گرافیکی ..	مبتنی بر رویداد .....
ک		open-source .....
Client .....	کارخواه .....	مدار مجتمع (IC) .....
Server .....	کارساز .....	مولار .....
MQTT Broker .....	کارگزار MQTT .....	مشترک .....
message broker .....	کارگزار پیام .....	معکوس .....
Access Control .....	کنترل دسترسی .....	مقیاس‌پذیر .....
گ	کنترل کننده‌های منطقی قابل برنامه‌ریزی Programmable Logic Controllers (PLCs)	مهاجم .....
asynchronous	گذرگاه ارتباطی سریال ناهمزمان serial communication bus	موضوع .....
		میان‌گیر .....
Bus .....	گذرگاه مشترک .....	ن
		ناشر .....
		baud rate .....
		Breadboard .....
		نگاره .....

# واژه‌نامه‌ی انگلیسی به فارسی

A

Access Control ..... کنترل دسترسی

Adversary ..... مهاجم

Arduino ..... آردوینو

asynchronous ..... درواهی ارتباطی مرکزی  
serial communication bus

Authentication ..... احراز هویت

B

Baudrate ..... نرخ باد

Breadboard ..... نگاره

Buffer ..... میان‌گیر

Bus ..... گذرگاه مشترک

Cable ..... ریسمان

Central ..... درواهی ارتباطی مرکزی

Communication Gateway

Chip ..... تراشه

Client ..... کارخواه

common-mode noise ..... اختلال حالت مشترک

Consistency ..... سازگاری

Customization ..... سفارشی‌سازی

D

Database ..... پایگاه داده

Debug ..... اشکال‌زدایی

Differential signalling ..... سیگنال‌دهی تفاضلی

Driver ..... راهانداز

E

Electronic ..... اتحادیه صنایع الکترونیک

Industries Alliance

Ethernet Port .....	درگاه اernetes .....	Media Access Control .....	آدرس MAC
Event .....	رویداد .....	Address (MAC Address)	
event driven .....	مبتنی بر رویداد .....	Modular .....	مدولار یا پیمانه‌ای .....
G		N	
GPIO .....	پایه‌های ورودی و خروجی عمومی .....	Noise .....	اختلال .....
GPU .....	واحدهای پردازش گرافیکی .....	non-inverting .....	غیرمعکوس .....
Gateway .....	درواره .....	O	
I		open-source .....	متن‌باز .....
IP .....	آی‌پی .....	Overlap .....	همپوشانی .....
Impedance matching .....	تطبیق امپدانس .....	P	
Integrated Circuit (IC) .....	مدار مجتمع .....	Pin .....	پایه .....
Internet of Things (IoT) .....	اینترنت اشیا .....	Port .....	درگاه .....
Inverting .....	معکوس .....	Portable .....	قابل حمل .....
L		Power .....	باتری .....
Lcd .....	صفحه نمایش .....	کنترل‌کننده‌های منطقی قابل برنامه‌ریزی	
M		Programmable Logic Controllers (PLCs)	
MQTT Broker .....	کارگزار MQTT .....	Publish .....	انتشار .....
Master .....	ارباب .....	Publisher .....	ناشر .....
message broker .....	کارگزار پیام .....	R	
		RAM .....	حافظه .....
		Raspberry Pi .....	رزبری‌پای .....

Recommended . . . . .	استاندارد توصیه شده . . . . .	خانه‌ی هوشمند . . . . .
	Standard	سیستم اندازه‌گیری هوشمند
Register . . . . .	ثبت . . . . .	System
Remote Terminal . . . . .	واحد ترمینال از راه دور . . . . .	قابل انباشتہ . . . . .
	Unit (RTU)	اشتراک . . . . .
Robustness . . . . .	استحکام . . . . .	مشترک . . . . .
S		سیستم روی تراشه . . . . .
Scalable . . . . .	مقیاس پذیر . . . . .	T
Schematic . . . . .	طرح واره . . . . .	آزمون . . . . .
Server . . . . .	کارساز . . . . .	موضوع . . . . .
Simulated . . . . .	شبیه سازی شده . . . . .	عیب یابی . . . . .
single-ended . . . . .	تک پایانی . . . . .	U
Slave . . . . .	برده . . . . .	اعتبارنامه های منحصر به فرد . . . . .
Smart Grid . . . . .	شبکه‌ی هوشمند . . . . .	Credentials