

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



گزارش نهایی پروژه سیستم‌های سایبر-فیزیکی عنوان پروژه

: گروه

علی امام زاده ، محمد حسین عقیلی ، علی بنی هاشمی ، محمد صادق عقیلی

شماره دانشجویی:

۸۱۰۱۰۰۲۷۴|۸۱۰۱۰۰۲۴۵

۸۱۰۱۹۹۳۷۷|۸۱۰۱۹۹۵۷۶

: تاریخ

۱۴۰۴ مرداد

استاد: مدرسی

۱۴۰۴-۱۴۰۳

فهرست مطالب

Error! Bookmark not defined.....	-1
Error! Bookmark not defined.....	-2
Error! Bookmark not defined.....	-3
Error! Bookmark not defined.....	-4
Error! Bookmark not defined.....	-5
Error! Bookmark not defined.....	-6
Error! Bookmark not defined.....	-7
Error! Bookmark not defined.....	-8
Error! Bookmark not defined.....	-9

۱. مقدمه

این گزارش به تشریح کامل مراحل طراحی، پیاده‌سازی و ارزیابی یک سیستم سایبر فیزیکی برای ردیابی و نمایش موقعیت جغرافیایی به صورت بلاذرنگ می‌پردازد. این پروژه در چارچوب درس «سیستم‌های سایبر فیزیکی» در دانشکده مهندسی برق و کامپیوتر دانشگاه تهران تعریف و اجرا شده است.

۱.۱. محدوده پروژه

محدوده این پروژه، طراحی و ساخت یک سیستم کامل و مستقل است که وظیفه اصلی آن، دریافت داده‌های موقعیت جغرافیایی از طریق یک مژول GPS و نمایش آنی این اطلاعات بر روی یک رابط کاربری گرافیکی مجهز به نقشه آفلاین می‌باشد. یکی از مشخصه‌های اصلی و محدودیت‌های طراحی این سیستم، استقلال عملکردی آن از شبکه اینترنت است. این ویژگی، به خصوص در بخش نمایش نقشه، سیستم را از نمونه‌های متداول که به سرویس‌های نقشه آنلاین (مانند Google Maps API) وابسته‌اند، تمایز می‌سازد و آن را برای کاربردهایی که در آن دسترسی به اینترنت پایدار یا ممکن نیست، مناسب می‌سازد.

مرزهای سیستم تعریف شده شامل چهار لایه اصلی است:

۱. **لایه حسگری فیزیکی**: متشکل از مژول SIMCom SIM808 که وظیفه ارتباط با ماهواره‌های سامانه موقعیت‌یاب جهانی (GPS) و استخراج داده‌های خام موقعیت را بر عهده دارد.
۲. **لایه پردازش تعبیه شده**: متشکل از برد میکروکنترلر Arduino Uno که فیرمور اصلی سیستم بر روی آن اجرا می‌شود. این لایه مسئولیت پردازش اولیه داده‌ها و مدیریت ارتباط با لایه‌های دیگر را دارد.
۳. **لایه واسط و انتقال داده**: این لایه بر روی یک کامپیوتر میزبان پیاده‌سازی شده و از طریق یک اسکریپت پایتون، داده‌های دریافتی از لایه تعبیه شده را برای لایه نمایش آماده می‌سازد.
۴. **لایه نمایش و تعامل**: شامل یک رابط کاربری مبتنی بر وب است که با استفاده از فناوری‌های استاندارد، موقعیت دریافت شده را به صورت یک نشانگر متحرک بر روی نقشه نمایش می‌دهد.

۱.۲. اهداف پروژه

اهداف اصلی این پروژه با تمرکز بر مفاهیم کلیدی سیستم‌های سایبر-فیزیکی و با توجه به نیازمندی‌های تعریف شده در فاز اولیه، به شرح زیر تدوین گردید:

- **هدف ۱: پیاده‌سازی یک حلقه کامل سایبر فیزیکی**: طراحی و ساخت یک سیستم یکپارچه که چرخه کامل «حسگری» (Sensing) پردازش (Computation) نمایش (Presentation) را پیاده‌سازی کند. این هدف، هسته اصلی پروژه و منطبق بر اصول بنیادی سیستم‌های سایبر فیزیکی است.

- هدف ۲: دریافت و پردازش داده‌های راهاندازی موفق ماژول SIM808 به منظور دریافت مستمر جملات استاندارد NMEA از ماهواره‌ها. این فرآیند شامل استخراج دقیق مختصات طول و عرض جغرافیایی از این جملات و تبدیل آن‌ها به فرمت قابل استفاده برای سیستم‌های نقشه‌یابی است.
- هدف ۳: نمایش بلاذرنگ موقعیت: ایجاد یک رابط کاربری گرافیکی (GUI) پویا و واکنش‌گرا که موقعیت مکانی را به صورت یک نشانگر متحرک بر روی نقشه نمایش دهد. این نمایش باید با کمترین تأخیر ممکن نسبت به داده‌های دریافتی از حسگر صورت پذیرد تا حس بلادرنگ بودن را به کاربر القا کند.
- هدف ۴: استقلال از شبکه اینترنت: حصول اطمینان از اینکه عملکرد اصلی سیستم، به ویژه بخش نمایش نقشه، هیچ‌گونه وابستگی به اتصال اینترنت نداشته باشد. این هدف از طریق پیش‌پردازش و ذخیره‌سازی محلی کاشی‌های نقشه (Map Tiles) محقق شده است.

۲. معرفی پلتفرم و ابزارهای استفاده شده

انتخاب دقیق ابزارها و پلتفرم‌های سخت‌افزاری و نرم‌افزاری، نقشی حیاتی در موفقیت این پروژه ایفا کرده است. معماری سیستم به گونه‌ای طراحی شده که هر جزء وظیفه مشخصی را بر عهده دارد و این تقسیم وظایف، توسعه و اشکال‌زدایی را تسهیل نموده است. جدول زیر، فهرست کاملی از مولفه‌های به کار رفته در این پروژه را به همراه نقش هر یک ارائه می‌دهد.

فهرست کامل سخت‌افزار و نرم‌افزار پروژه:

مولفه/ابزار	شرح و نقش در پروژه
Arduino Uno	میکروکنترلر مرکزی سیستم که به عنوان مغز متفکر لایه تعییه‌شده عمل می‌کند. وظیفه اجرای فیرمور(GpsSensor.ino)، برقراری ارتباط با ماژول GPS از طریق پورت سریال نرم‌افزاری، پars کردن جملات NMEA و ارسال مختصات نهایی از طریق پورت سریال USB به کامپیوتر میزبان را بر عهده دارد (عکس ۱).
SIMCom SIM808	مولفه اصلی حسگری پروژه که دارای یک گیرنده GPS داخلی است. در این پروژه، از قابلیت GPS این ماژول برای دریافت داده‌های موقعیت از ماهواره‌ها استفاده شده است. این ماژول داده‌ها را در قالب استاندارد NMEA از طریق پین TX خود ارسال می‌کند. (عکس ۱)
کامپیوتر میزبان	یک لپتاپ (بر اساس شناسه پورت /dev/cu.usbserial-10) که در عکس ۴ قابل مشاهده است.
Arduino IDE	محیط توسعه یکپارچه (IDE) که برای برنامه‌نویسی، کامپایل و بارگذاری کد به زبان C++ بر روی برد آردوینو استفاده شده است.
	زبان برنامه‌نویسی سطح بالا که برای توسعه اسکریپت سمت میزبان (savelocation.py) به کار رفته است. این اسکریپت به عنوان یک پل ارتباطی، داده‌ها را از پورت سریال خوانده و در قالبی قابل دسترس برای لایه نمایش ذخیره می‌کند.

مولفه/ابزار

شرح و نقش در پروژه

یک کتابخانه استاندارد در پایتون که امکان برقراری ارتباط دوطرفه با دستگاه‌های متصل به پورت‌های سریال را فراهم می‌آورد و در این پروژه برای خواندن داده از آردوبینو استفاده شده کتابخانه pySerial است.

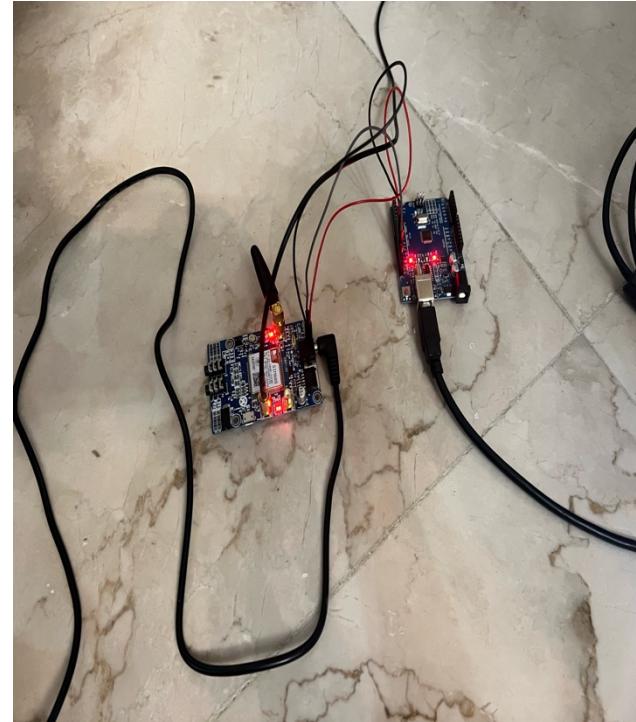
مجموعه‌ای از فناوری‌های استاندارد وب که برای ساختاردهی(HTML) ، استایل‌دهی (CSS) و ایجاد پویایی و تعامل (JavaScript) در رابط کاربری نقشه (map.html) به کار گرفته شده‌اند.[map.html]

یک کتابخانه جاوا اسکریپت متن‌باز، سیک و قدرتمند برای ایجاد نقشه‌های تعاملی. این کتابخانه به دلیل سادگی و عملکرد بالا برای نمایش کاشی‌های نقشه آفلاین و مدیریت نشانگر متحرک موقعیت انتخاب شده است.

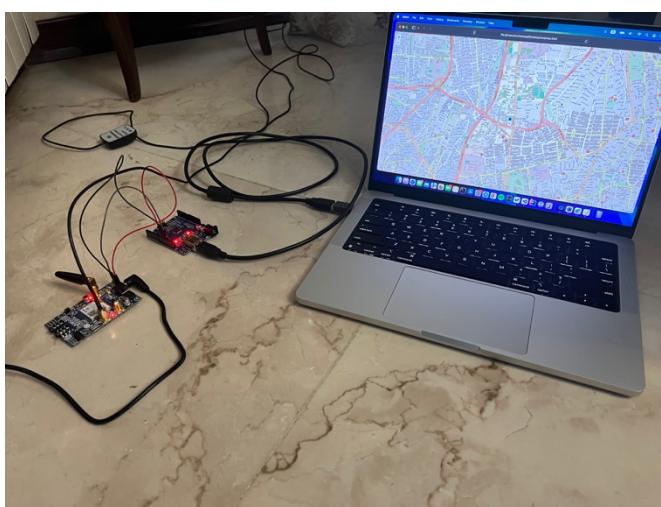
مجموعه‌ای از تصاویر نقشه (در فرمت PNG) که از قبل برای منطقه تهران از منابعی مانند OpenStreetMap استخراج و در ساختار پوشه‌ای مشخص به صورت محلی ذخیره شده‌اند. این رویکرد، هسته اصلی قابلیت آفلاین سیستم را تشکیل می‌دهد.



عکس ۲: انتن gps متصل به sim808



عکس ۱: اتصالات برد اردوینو و sim 808x



عکس ۴: کامپیوتر میزبان



عکس ۳: پاور متصل به sim 808

۳. تغییرات نسبت به فاز پروپوزال

۳.۱. چالش‌های پروژه

در طول اجرای پروژه، تیم با سه چالش عمدۀ مواجه شد که مدیریت آن‌ها نیازمند صرف زمان و منابع بود:

- چالش ۱: محدودیت سخت‌افزاری اولیه: در ابتدای پروژه، مازول SIM900 برای دریافت داده‌ها در نظر گرفته شده بود. اما پس از تلاش‌های اولیه، مشخص شد که این مازول قادر گیرنده GPS داخلی است و تنها قابلیت‌های GSM/GPRS را فراهم می‌کند. این عدم تطابق با هدف اصلی پروژه، تیم را با یک مانع اساسی روبرو کرد.
- چالش ۲: تهیی و مدیریت نقشه آفلاین: یکی از اهداف کلیدی پروژه، استقلال از اینترنت بود که نیازمند تهیی نقشه‌های آفلاین بود. فرآیند یافتن، دانلود و ساختاردهی کاشی‌های نقشه (raster tiles) برای شهر تهران در سطوح زوم مختلف، فرآیندی پیچیده و زمانبر بود. علاوه بر این، حجم بالای این کاشی‌ها (به خصوص در سطوح زوم بالا) یک چالش لجستیکی در مدیریت و توزیع فایل‌های پروژه محسوب می‌شد.
- چالش ۳: دریافت مختصات ۰,۰ : در مراحل ابتدایی، مازول SIM808 علی‌رغم اجرای صحیح کد، مختصات (۰,۰, ۰,۰) را بازمی‌گرداند و چراغ جی بی اس ان بصورت چشمکزن تند بود و این به معنای متصل نشدن انتن به ماهواره است.
(همچنین به دلایل شرایط پیش امده در کشور اختلالات ماهواره‌ای GPS در تهران کار ما را تا چند روزی مختل کرد ولی خوشبختانه این موضوع حل شد

راه حل: تست در فضای باز و حداقل ۱ ساعت زمان دادن برای کانکت شدن مازول به ماهواره.

۳.۲. دلایلی که منجر به تغییر در تصمیم‌گیری‌ها شدند

چالش‌های ذکر شده، تیم را به سمت اتخاذ تصمیمات اصلاحی مهمی سوق داد:

- تصمیم ۱: مهاجرت از SIM900 به SIM808: این مهمترین و حیاتی‌ترین تغییر در پروژه بود. پس از مشخص شدن نقص مازول SIM900 برای اهداف پروژه، تیم به سرعت اقدام به جایگزینی آن با مازول SIM808 کرد که علاوه بر قابلیت‌های GSM، دارای یک گیرنده GPS داخلی نیز می‌باشد.
- تصمیم ۲: انتخاب پشتۀ فناوری HTML/Leaflet.js: در ابتدا، گزینه‌های مختلفی برای نمایش نقشه، از جمله نرم‌افزارهای تخصصی مانند OpenCPN، مورد بررسی قرار گرفتند. با این حال، تیم در نهایت پشتۀ

فناوری مبتنی بر وب شامل HTML و کتابخانه Leaflet.js را انتخاب کرد. دلایل اصلی این انتخاب، «سادگی، ماهیت آفلاین و عملکرد سریع» این راهکار بود.

۳.۳. تجارت ناموفق

استفاده از منبع تغذیه نا مناسب در ابتدا باعث شد انتن جی بی اس به درستی کار نکند. بعد از تحقیق به این نتیجه رسیدیم که اداپتور باید حداقل دارای ۲ امپر و بالای ۵ ولت باشد.

۴. نزدیک‌ترین نمونه‌های مشابه

- نمونه‌های تجاری: CalAmp ، Teltonikas (Commercial Fleet Tracking Systems) شرکت‌هایی مانند Arusnavi و GPS را از طریق شبکه GPRS به یک سرور ابری ارسال کرده و داشبوردهای تحلیلی پیچیده‌ای برای مانیتورینگ، گزارش‌گیری و بهینه‌سازی مسیرها فراهم می‌کنند. تفاوت بنیادین پروژه حاضر با این سیستم‌ها در معماری آن است. سیستم ما یک راهکار کاملاً آفلاین و محلی (local) است که برای مانیتورینگ از یک ایستگاه ثابت طراحی شده و به هیچ زیرساخت ابری یا اتصال اینترنت وابستگی ندارد. این ویزگی آن را برای کاربردهای آموزشی، تحقیقاتی یا سناریوهایی با محدودیت اتصال، ایده‌آل می‌سازد.

- پروژه‌های متن‌باز و Instructables (DIY Open-Source Projects): در پلتفرم‌هایی مانند GitHub و Hackaday، پروژه‌های ردیاب GPS متعددی با استفاده از آردوینو و ماژول‌های مشابه یافت می‌شود. با این حال، اکثر این پروژه‌ها یکی از دو رویکرد زیر را دنبال می‌کنند: ۱) ثبت داده‌ها بر روی یک کارت حافظه SD برای تحلیل پس از رویداد، یا ۲) ارسال داده‌ها از طریق بلوتوث (Logging) به یک اپلیکیشن اختصاصی موبایل. معماری این پروژه با معرفی یک لایه میانی (اسکریپت پایتون) و استفاده از یک فایل متنی به عنوان واسطه، رویکرد متفاوتی را ارائه می‌دهد. این معماری، سختافزار را به طور کامل از نرمافزار نمایش جدا می‌کند و انعطاف‌پذیری بالایی را فراهم می‌آورد. به عنوان مثال، می‌توان به سادگی رابط کاربری وب را با یک برنامه دسکتاپ نیتیو جایگزین کرد بدون آنکه نیازی به تغییر در کد آردوینو یا اسکریپت پایتون باشد.

۵. مبانی فنی پروژه

این بخش به تشریح عمیق معماری فنی سیستم، جریان داده‌ها از حسگر تا نمایشگر، و تحلیل عملکردی راه حل پیاده‌سازی شده می‌پردازد.

۵.۱. ارائه راه حل با جزئیات

جریان داده در سیستم به شرح زیر است:

1. دریافت سیگنال: مژول SIM808 سیگنال‌های زمان‌بندی شده را از حداقل چهار ماهواره GPS دریافت می‌کند.
2. محاسبه و خروجی: NMEA پردازنده داخلی مژول، بر اساس این سیگنال‌ها موقعیت خود را محاسبه کرده و نتیجه را در قالب جملات استاندارد (NMEA: پروتکل استاندارد ارتباطی برای تجهیزات دریایی) از طریق پین TX خود به صورت سریال ارسال می‌کند.
3. دریافت در آردوینو: برد آردوینو این جملات را از طریق یک پورت سریال نرمافزاری که بر روی پین‌های ۱۰ و ۱۱ تعریف شده (SoftwareSerial)، دریافت می‌کند. استفاده از سریال نرمافزاری این امکان را فراهم می‌کند که پورت سریال سخت‌افزاری اصلی برای ارتباط با کامپیوتر و دیباگینگ آزاد باقی بماند.
4. پars کردن در آردوینو: (GpsSensor.ino) جملات دریافتی را خط به خط می‌خواند و تنها جملاتی را که با \$GPGGA شروع می‌شوند، برای پردازش انتخاب می‌کند. این نوع جمله حاوی اطلاعات کلیدی موقعیت، زمان و کیفیت سیگنال (Fix data) است.
5. استخراج و تبدیل مختصات: کد آردوینو با پیمایش جمله \$GPGGA و جداسازی فیلدها بر اساس کاراکتر (,), مقادیر خام طول و عرض جغرافیایی را استخراج می‌کند. سپس تابع convert.ToDecimal این مقادیر را که در فرمت (DDMM.MMmm درجه و دقایق) هستند، به فرمت استاندارد درجه اعشاری (Decimal Degrees) تبدیل می‌کند که برای نرمافزارهای نقشه‌یابی قابل فهم است.
6. ارسال به کامپیوتر: مختصات پردازش شده به صورت یک رشته متنی ساده (مانند 35.786319, 51.454273) از طریق پورت سریال USB به کامپیوتر میزبان ارسال می‌شود.
7. خواندن از پورت سریال: (save_location.py) اسکریپت پایتون که بر روی کامپیوتر میزبان در حال اجراست، به طور مداوم پورت سریال را برای دریافت داده‌های جدید مانیتور می‌کند و هر خط دریافتی را می‌خواند.
8. ذخیره‌سازی در فایل واسط: اسکریپت پایتون پس از دریافت هر خط معتبر، فایل location.txt را در حالت نوشتن ("w") باز کرده و رشته مختصات را در آن می‌نویسد. استفاده از حالت "w" باعث می‌شود که با هر بروزرسانی، محتوای قبلی فایل به طور کامل پاک شده و با داده جدید جایگزین شود. این مکانیزم ساده، از رشد بی‌رویه فایل جلوگیری کرده و اطمینان می‌دهد که فایل همواره حاوی آخرین موقعیت معتبر است.
9. واکشی داده در رابط کاربری: (map.html) در سمت کلاینت (مروگر وب)، یک حلقه setInterval در کد جاوا اسکریپت، هر ۱۰۰۰ میلی‌ثانیه (۱ ثانیه) یک بار اجرا می‌شود.
10. درخواست HTTP محلی: در هر تکرار حلقه، یک درخواست fetch به فایل محلی location.txt ارسال می‌شود. برای جلوگیری از اینکه مروگر از نسخه کش شده فایل استفاده کند، یک پارامتر زمان (new)

به انتهای URL اضافه می‌شود. این تکنیک که به آن "cache busting" می‌گویند، Date().getTime()

تضمین می‌کند که مرورگر همواره جدیدترین نسخه فایل را از دیسک بخواند.

11. به روزرسانی نقشه: پس از دریافت محتوای فایل، کد جاوا اسکریپت آن را پارس کرده، مختصات را استخراج می‌کند و با استفاده از متده Leaflet.setLatLng، موقعیت نشانگر (marker) و هاله اطراف آن (halo) را بر روی نقشه به روزرسانی می‌کند. این عملیات بدون نیاز به بارگذاری مجدد کل صفحه انحصار می‌شود و تجربه‌ای روان و بلادرنگ را برای کاربر فراهم می‌آورد.

۵.۲. تحلیل راحل و اثبات کارایی

- **تأخیر:** عوامل اصلی تأخیر عبارتند از: فرکانس به روزرسانی ماژول GPS (که معمولاً ۱ هرتز یا ۱ ثانیه است)، تأخیر پردازش در آردوبینو (ناچیز، در حد میکروثانیه)، تأخیر انتقال سریال (وابسته به نرخ باود ۹۶۰۰)، و فاصله زمانی نظرسنجی (polling interval) در جاوا اسکریپت (تنظیم شده بر روی ۱۰۰۰ میلیثانیه). با توجه به این موارد، گلوگاه اصلی تأخیر در سیستم، فرکانس به روزرسانی GPS و بازه زمانی setInterval است که هر دو بر روی ۱ ثانیه تنظیم شده‌اند. بنابراین، تأخیر کلی سیستم در حدود ۱ تا ۲ ثانیه تخمین زده می‌شود.

- **دقت:** دقت (Precision vs. Accuracy) کد آردوبینو (GpsSensor.ino) مختصات را با ۶ رقم اعشار ارسال می‌کند. این سطح از دقت عددی (precision) در تئوری معادل تفکیک‌پذیری حدود ۱۱ سانتی‌متر در خط استوا است. با این حال، این دقت عددی با دقت واقعی (accuracy) سیستم متفاوت است. دقت واقعی یک گیرنده GPS تجاری مانند SIM808 در شرایط بهینه (فضای باز با دید خوب به آسمان) معمولاً در محدوده ۲/۵ تا ۱۰ متر قرار دارد. بنابراین، اگرچه سیستم مختصات را با ارقام اعشار بالا نمایش می‌دهد، خطای واقعی موقعیت‌یابی بیشتر از آن است.

- **صرف منابع:** مصرف منابع (Resource Consumption) کد آردوبینو به دلیل سادگی و عدم استفاده از کتابخانه‌های سنگین، مصرف بسیار پایینی از حافظه فلاش و RAM میکروکنترلر دارد. در سمت کامپیوتر میزبان نیز، اسکریپت پایتون به دلیل ماهیت ورودی/خروجی محور خود، بار پردازشی ناچیزی بر روی CPU دارد. رابط کاربری وب نیز به لطف سبکی کتابخانه Leaflet.js و سادگی عملیات رندرینگ، منابع کمی از مرورگر را اشغال می‌کند و عملکردی روان را حتی بر روی سخت‌افزارهای ضعیفتر تضمین می‌نماید.

۶. جزئیات پیاده‌سازی

۶.۱. شکست کار بین اعضای تیم

موفقیت پروژه حاصل کار گروهی و تقسیم وظایف دقیق بین اعضای تیم بود. مسئولیت هر یک از بخش‌های اصلی پروژه به شرح زیر تقسیم شد:

- علی بنی هاشمی : مسئولیت بخش سخت افزار پروژه، شامل سیم کشی فیزیکی بین برد آردوینو و ماژول SIM808، تأمین تغذیه مناسب برای ماژول (که به جریان بالاتری نسبت به خروجی پین های آردوینو نیاز دارد) و اطمینان از پایداری اتصالات فیزیکی.
- علی امام زاده : مسئولیت توسعه فیرمور آردوینو (GpsSensor.ino) این وظیفه شامل پیاده سازی منطق ارتباط سریال با ماژول GPS، نوشتن الگوریتم کارآمد برای پars کردن جملات NMEA و تابع تبدیل فرمت مختصات جغرافیایی بود.
- محمد صادق عقیلی : مسئولیت طراحی و پیاده سازی کامل سیستم نقشه آفلاین. این بخش شامل تحقیق در مورد فرمتهای نقشه، تهیه کاشهای نقشه (raster tiles) برای منطقه مورد نظر و کدنویسی فایل map.html با استفاده از کتابخانه Leaflet.js برای نمایش نقشه و نشانگر موقعیت بود.
- محمد حسین عقیلی : مسئولیت ایجاد ارتباط بین سخت افزار و نرم افزار نمایش. این وظیفه شامل توسعه اسکریپت پایتون (save_location.py) برای خواندن داده از پورت سریال آردوینو و پیاده سازی مکانیزم ذخیره سازی داده در فایل واسط location.txt بود.

۴.۲. مشخصات محیط توسعه

توسعه پروژه بر روی پلتفرمها و ابزارهای زیر انجام گرفت:

- سیستم عامل: macOS
- محیط توسعه آردوینو IDE: نسخه 2.3.7 Arduino IDE (nightly)
- پایتون: نسخه 3
- مرورگر وب: هر مرورگر مدرن با پشتیبانی از HTML5 و Fetch API ، JavaScript ES6 مانند (Google Chrome, Safari).

۴.۳. تشریح پیاده سازی

الف) سخت افزار: اتصالات فیزیکی بین برد آردوینو Uno و ماژول SIM808 . شماتیک اتصالات به شرح زیر است:

- پین خروجی سریال ماژول (TXD) به پین ورودی سریال نرم افزاری آردوینو (پین دیجیتال ۱۰) متصل شد.
- پین ورودی سریال ماژول (RXD) به پین خروجی سریال نرم افزاری آردوینو (پین دیجیتال ۱۱) متصل شد.
- پین زمین ماژول (GND) به یکی از پین های زمین (GND) آردوینو متصل شد.
- تغذیه ماژول (VCC) به یک منبع تغذیه خارجی مناسب (که قادر به تأمین جریان پیک مورد نیاز ماژول باشد) متصل گردید، زیرا پین های خروجی آردوینو توانایی تأمین این جریان را ندارند.

ب) فیرمور آردوینو: کد آردوینو وظیفه پردازش اولیه و حیاتی داده ها را بر عهده دارد.

توضیحات کد:

```
#include <SoftwareSerial.h>
```

این خط، کتابخانه SoftwareSerial را به پروژه اضافه می‌کند که امکان شبیه‌سازی یک پورت سریال بر روی هر دو پین دیجیتال آردوبینو را فراهم می‌آورد.

```
SoftwareSerial gpsSerial(10, 11)
```

یک نمونه از پورت سریال نرم‌افزاری با نام gpsSerial ایجاد می‌شود که از پین ۱۰ به عنوان RX برای دریافت داده از SIM808 و از پین ۱۱ به عنوان TX برای ارسال داده به SIM808 استفاده می‌کند.

منطق حلقه اصلی : در حلقه اصلی، کد به طور مداوم بررسی می‌کند که آیا داده‌ای در بافر سریال نرم‌افزاری (gpsSerial) موجود است یا خیر. در صورت وجود، کاراکترها را یکی‌یکی می‌خواند و به یک رشته (nmeaLine) اضافه می‌کند تا زمانی که به کاراکتر خط جدید (\n) برسد. این روش، تضمین می‌کند که همواره یک جمله کامل NMEA پردازش می‌شود.

```
if (nmeaLine.startsWith("$GPGGA")):
```

پس از دریافت یک خط کامل، کد بررسی می‌کند که آیا آن خط با شناسه \$GPGGA شروع می‌شود یا خیر. این فیلتر، از پردازش اطلاعات غیرضروری از انواع دیگر جملات NMEA جلوگیری کرده و منابع پردازشی را بهینه می‌کند.

```
parseGPGGA(String sentence)
```

این تابع، منطق اصلی پars کردن را در خود جای داده است. با پیمایش کاراکتر به کاراکتر رشته و جستجو برای جداکننده (',')، جمله را به فیلدهای مجزا تقسیم کرده و در یک آرایه از رشته‌ها ذخیره می‌کند.

```
convert.ToDecimal(float raw, String direction):
```

این تابع داده‌های طول و عرض جغرافیایی در فرمت NMEA به صورت DDMM.MMMM به فرمت Decimal Degrees (Decimal Degrees) که برای کتابخانه‌هایی مانند Leaflet قابل استفاده است، تبدیل می‌کند.

ج) اسکریپت واسط پایتون (save-location.py)
توضیحات:

```
:import serial
```

کتابخانه pySerial که ابزار اصلی برای کار با پورت‌های سریال در پایتون است، وارد می‌شود.

```
: ser = serial.Serial("/dev/cu.usbserial-10", 9600)
```

یک شیء از کلاس Serial ایجاد می‌شود. پارامتر اول، آدرس پورت سریال است که آردوبینو به آن متصل است (این آدرس در سیستم عامل‌های مختلف متفاوت است). پارامتر دوم، نرخ باود (Baud Rate) است که باید دقیقاً با مقدار تنظیم شده در کد آردوبینو (Serial.begin(9600)) یکسان باشد.

```
: line = ser.readline().decode().strip()
```

ser.readline() منتظر می‌ماند تا یک خط کامل تا کاراکتر \n از پورت سریال دریافت شود

decode بایت‌های دریافتی را به یک رشته پایتون تبدیل می‌کند

strip() هرگونه فضای خالی یا کاراکتر خط جدید اضافی را از ابتدا و انتهای رشته حذف می‌کند.

```
: with open("../location.txt", "w") as f
```

این دستور، فایل location.txt را در حالت نوشتن ("w") باز می‌کند.

۵) رابط کاربری نقشه (map.html): این فایل وظیفه نمایش داده‌ها و رابط تعاملی کاربر را دارد.

توضیحات:

```
L.tileLayer('tiles/{z}/{x}/{y}.png')
```

این خط به کتابخانه Leaflet دستور می‌دهد که کاشی‌های نقشه را از یک مسیر محلی (/tiles/) بارگذاری کند. متغیرهای {z} سطح زوم ({x} و {y} مختصات کاشی) توسط Leaflet به صورت خودکار جایگزین می‌شوند تا تصویر مناسب برای هر قسمت از نقشه بارگذاری شود. این نشان دهنده قابلیت آفلاین سیستم است.

```
setInterval(() => { ... }, 1000)
```

این دستور یک تابع را به صورت متناوب هر ۱۰۰۰ میلی‌ثانیه یک بار اجرا می‌کند.

```
fetch('location.txt?' + new Date().getTime()):
```

تابع fetch برای خواندن محتوای فایل location.txt استفاده می‌شود

```
marker.setLatLng(position):
```

پس از واکشی و پارس کردن موفقیت‌آمیز مختصات، این متده موقعیت نشانگر (marker) را بر روی نقشه به

صورت پویا و بدون نیاز به بارگذاری مجدد کل صفحه، به روزرسانی می‌کند. این امر منجر به ایجاد یک اینیمیشن روان از حرکت نشانگر بر روی نقشه می‌شود.

۷. آزمون، ارزیابی و مقایسه عملکرد

برای اطمینان از صحت عملکرد و کارایی سیستم، مجموعه‌های از آزمون‌های جامع طراحی و اجرا گردید. این بخش به تشریح طرح آزمون، نحوه اجرا و تحلیل نتایج به دست آمده می‌پردازد.

۷.۱. طرح آزمون

طرح آزمون برای پوشش دادن تمام جنبه‌های عملکردی سیستم، از دریافت داده تا نمایش نهایی، تدوین شد:

- آزمون ۱: صحت‌سنجی خروجی سریال آردوبینو: هدف این آزمون، تأیید این موضوع بود که فیرمور آردوبینو به درستی جملات NMEA را پارس کرده و مختصات را در فرمت صحیح (دو عدد اعشاری جدا شده با کاما) و با مقادیر منطقی از طریق پورت سریال ارسال می‌کند. این آزمون با استفاده از سریال مانیتور در Arduino IDE انجام شد.
- آزمون ۲: صحت‌سنجی عملکرد فایل واسط: هدف این آزمون، بررسی این بود که آیا اسکریپت پایتون (به درستی داده‌ها را از پورت سریال می‌خواند و فایل location.txt را به صورت مدام و صحیح با آخرین مختصات دریافتی به روزرسانی می‌کند.
- آزمون ۳: صحت‌سنجی نمایشگر نقشه: این آزمون به صورت بصری انجام شد و هدف آن تأیید حرکت صحیح و روان نشانگر بر روی نقشه آفلاین بود. موقعیت نمایش داده شده بر روی نقشه با موقعیت فیزیکی واقعی دستگاه (با در نظر گرفتن خطای ذاتی GPS) مقایسه شد.
- آزمون ۴: اندازه‌گیری زمان تا اولین موقعیت‌یابی (Time to First Fix – TTFF) این یک متريک استاندارد برای ارزیابی عملکرد گيرنده‌های GPS است. در این آزمون، زمان سپری شده از لحظه روشن شدن سیستم تا زمانی که اولین مختصات معتبر در خروجی سریال ظاهر می‌شود، اندازه‌گیری شد. این آزمون در شرایط شروع سرد (Cold Start) انجام گرفت.
- آزمون ۵: تست پایداری (Stability Test): برای اطمینان از عدم وجود نشت حافظه (memory leak) یا بروز خطاهای پیش‌بینی‌نشده در درازمدت، سیستم برای یک دوره طولانی (۱ ساعت) به صورت مدام روشن گذاشته شد و عملکرد تمام اجزای آن تحت نظر قرار گرفت.

۷.۲. نحوه اجرای آزمون

آزمون‌های عملکردی (به ویژه آزمون‌های ۳ و ۴) در یک فضای باز در محوطه دانشگاه تهران انجام شدند تا از دریافت سیگنال قوی و بدون مانع از ماهواره‌های GPS اطمینان حاصل شود. در طول تست، برد آردوبینو از طریق کابل USB به کامپیوتر میزبان متصل بود و هر سه جزء اصلی سیستم (فیرمور آردوبینو، اسکریپت پایتون و صفحه وب در مرورگر) به طور همزمان در حال اجرا بودند.

۷.۳ نتایج آزمون‌های انخام شده

نتایج به دست آمده از آزمون‌ها، موفقیت‌آمیز بودن عملکرد سیستم را تأیید کردند.

- خروجی سریال : خروجی سریال مانیتور به وضوح نشان داد که کد آردوینو با موفقیت داده‌های NMEA را به مختصات طول و عرض جغرافیایی تبدیل می‌کند



```

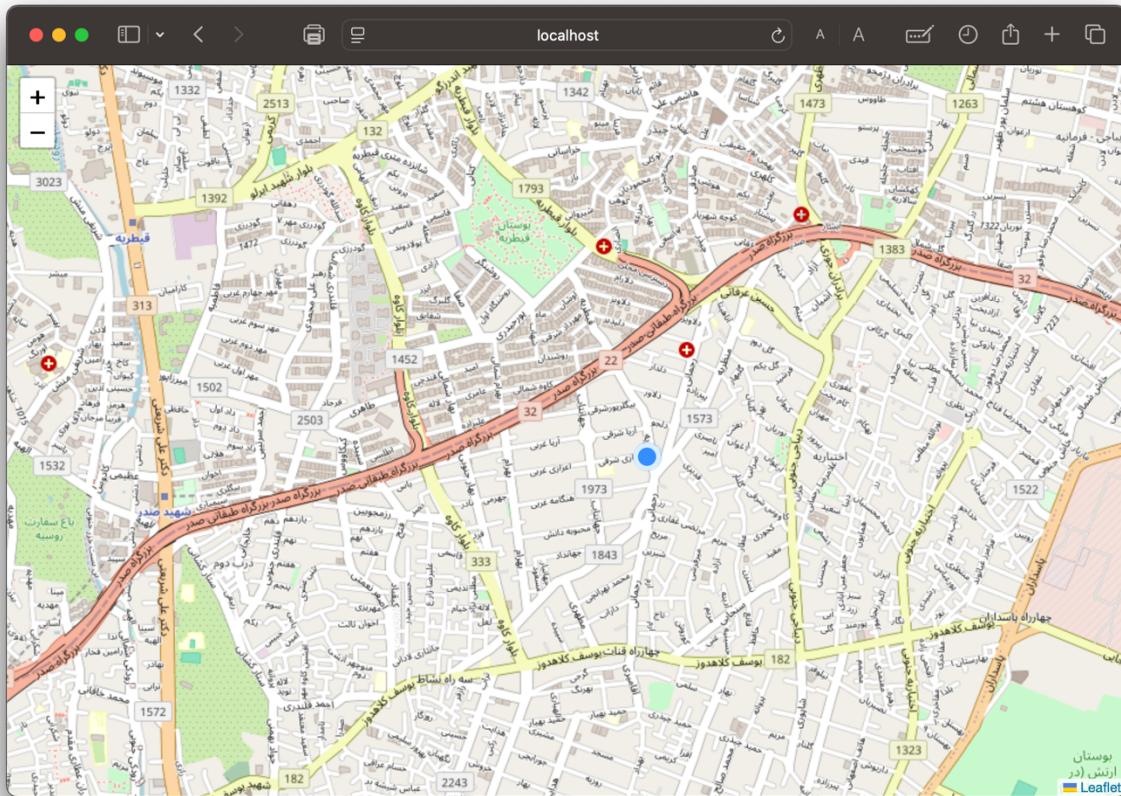
Output  Serial Monitor X
Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbserial-10')
Latitude: 35.786403
L Latitude: 35.786403
Longitude: 51.454280
-----
Latitude: 35.786415
Longitude: 51.454280
-----
Latitude: 35.786415
Longitude: 51.454280
-----
Latitude: 35.786415
Longitude: 51.454280
-----
```

Latitude: 35.786403

Longitude: 51.454208

تحلیل خروجی نشان می‌دهد که مقادیر در محدوده جغرافیایی تهران قرار دارند و فرمت آن‌ها برای پردازش توسط اسکریپت پایتون مناسب است.

- **نایشگر نقشه** : رابط کاربری نقشه به درستی بارگذاری شد و پس از چند ثانیه (زمان لازم برای TTFF)، نشانگر موقعیت بر روی نقشه ظاهر شد و با حرکت فیزیکی دستگاه، به نرمی جابجا می‌شد.



- نتایج عملکردی: نتایج کمی آزمون‌های عملکردی در جدول زیر خلاصه شده است.

جدول ۲: خلاصه نتایج آزمون‌های عملکردی

توضیحات:

زمان متصل شدن و فکیس شدن GPS سنسور با توجه به شرایط کنونی کشور و محدودیت های اعمال شده متغیر هست و از چند ثانیه تا چند ساعت ممکن است طول بکشد که بستگی به منطقه و شرایط آنتن دهی GPS در آن مکان دارد.

زمان تا اولین موقعیت یابی (TTFF) این زمان در شرایط شروع سرد (Cold Start) و در فضای باز اندازه‌گیری شده است. در حدود ۲ دقیقه شروع‌های گرم (Warm Start) این زمان به طور قابل توجهی کمتر بود.

این دقت با مقایسه موقعیت نمایش داده شده با موقعیت اصلی شخص تست کننده به عنوان مرجع، تخمین زده شده است.

این تأخیر مطابق با تنظیمات ۱ هرتزی مژول GPS و بازه زمانی ۱ ثانیه‌ای setInterval در ۱ ثانیه سرتاسری کد جاوا اسکریپت است.

در طول تست یک ساعته، هیچ‌گونه خطا، توقف ناگهانی یا کاهش عملکردی در هیچ‌یک از پایداری در اجزای سیستم مشاهده نشد.

۷/۴. تحلیل نتایج و مقایسه

نتایج به دست آمده به طور کامل با اهداف اولیه پروژه مطابقت دارند و کارایی و پایداری راه حل پیاده‌سازی شده را اثبات می‌کنند. تأخیر به روزرسانی ۱ ثانیه‌ای، تجربه‌ای نزدیک به بلادرنگ را برای کاربر فراهم می‌کند و برای اکثر کاربردهای نظارتی و ناوبری عمومی کاملاً مناسب است. دقت مکانی به دست آمده نیز در محدوده مورد انتظار برای گیرنده‌های GPS تجاری سطح مصرف کننده قرار دارد و برای این پروژه کافی است.

در مقایسه با نمونه‌های مشابه، سیستم طراحی شده از نظر عملکرد پایه (دقت و تأخیر) با بسیاری از پروژه‌های DIY برابر می‌کند. با این حال، مزیت اصلی آن در معماری مژولار، استفاده هوشمندانه از یک فایل واسطه برای جداسازی لایه‌ها و مهم‌تر از همه، قابلیت عملکرد کاملاً آفلاین است که آن را از بسیاری از راه حل‌های دیگر متمایز می‌سازد.

۸. پیوست‌های فنی

• پیوست ۱: کد کامل فیرمور آردوینو (GpsSensor.ino)

```
C++#include <SoftwareSerial.h>
SoftwareSerial gpsSerial(10, 11); // RX, TX
```

```
String nmeaLine = "";  
  
void setup() {  
    Serial.begin(9600);  
    gpsSerial.begin(9600);  
    Serial.println("Reading GPS data...");  
}  
  
void loop() {  
    while (gpsSerial.available()) {  
        char c = gpsSerial.read();  
        if (c == '\n') {  
            if (nmeaLine.startsWith("$GPGGA")) {  
                parseGPGGA(nmeaLine);  
            }  
            nmeaLine = "";  
        } else {  
            nmeaLine += c;  
        }  
    }  
    delay(200);  
}  
  
void parseGPGGA(String sentence) {  
    int commaIndex = 0;  
    int fieldIndex = 0;  
    String fields[15];  
  
    for (int i = 0; i < sentence.length(); i++) {  
        if (sentence.charAt(i) == ',' || sentence.charAt(i) == '*') {  
            fields[fieldIndex++] = sentence.substring(commaIndex, i);  
            commaIndex = i + 1;  
        }  
    }  
  
    if (fields[2].length() > 0 && fields[4].length() > 0) {  
        float lat = convertToDecimal(fields[2].toFloat(), fields[3]);  
        float lon = convertToDecimal(fields[4].toFloat(), fields[5]);  
  
        Serial.print(lat, 6);  
        Serial.print(",");  
        Serial.println(lon, 6);  
    }  
}  
  
float convertToDecimal(float raw, String direction) {  
    int degrees = int(raw / 100);  
    float minutes = raw - (degrees * 100);
```

```

    float decimal = degrees + (minutes / 60.0);
    if (direction == "S" || direction == "W") decimal *= -1;
    return decimal;
}

```

پیوست ۲: کد کامل اسکریپت پایتون (save-location.py)

```

#include <SoftwareSerial.h>
SoftwareSerial gpsSerial(10, 11); // RX, TX

String nmeaLine = "";

void setup() {
    Serial.begin(9600);
    gpsSerial.begin(9600);
    Serial.println("Reading GPS data...");
}

void loop() {
    while (gpsSerial.available()) {
        char c = gpsSerial.read();
        if (c == '\n') {
            if (nmeaLine.startsWith("$GPGGA")) {
                parseGPGGA(nmeaLine);
            }
            nmeaLine = "";
        } else {
            nmeaLine += c;
        }
    }
    delay(200);
}

void parseGPGGA(String sentence) {
    int commaIndex = 0;
    int fieldIndex = 0;
    String fields[15];

    for (int i = 0; i < sentence.length(); i++) {
        if (sentence.charAt(i) == ',' || sentence.charAt(i) == '*') {
            fields[fieldIndex++] = sentence.substring(commaIndex, i);
            commaIndex = i + 1;
        }
    }

    if (fields[2].length() > 0 && fields[4].length() > 0) {
        float lat = convert.ToDecimal(fields[2].toFloat(), fields[3]);
    }
}

```

```

    float lon = convertToDecimal(fields[4].toFloat(), fields[5]);

    Serial.print(lat, 6);
    Serial.print(",");
    Serial.println(lon, 6);
}
}

float convertToDecimal(float raw, String direction) {
    int degrees = int(raw / 100);
    float minutes = raw - (degrees * 100);
    float decimal = degrees + (minutes / 60.0);
    if (direction == "S" || direction == "W") decimal *= -1;
    return decimal;
}

```

پیوست ج: کد کامل رابط کاربری نقشه (map.html) •

```

<html>
  <head>
    <meta charset="utf-8" />
    <title>Tehran Offline Map</title>
    <link
      rel="stylesheet"
      href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
    />
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
    <style>
      html,
      body,
      #map {
        height: 100%;
        margin: 0;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
      const map = L.map("map").setView([35.724688, 51.387633], 15);

      L.tileLayer("tiles/{z}/{x}/{y}.png", {
        minZoom: 5,
        maxZoom: 18,
        attribution: ""
      })
    </script>
  </body>
</html>

```

```
}).addTo(map);

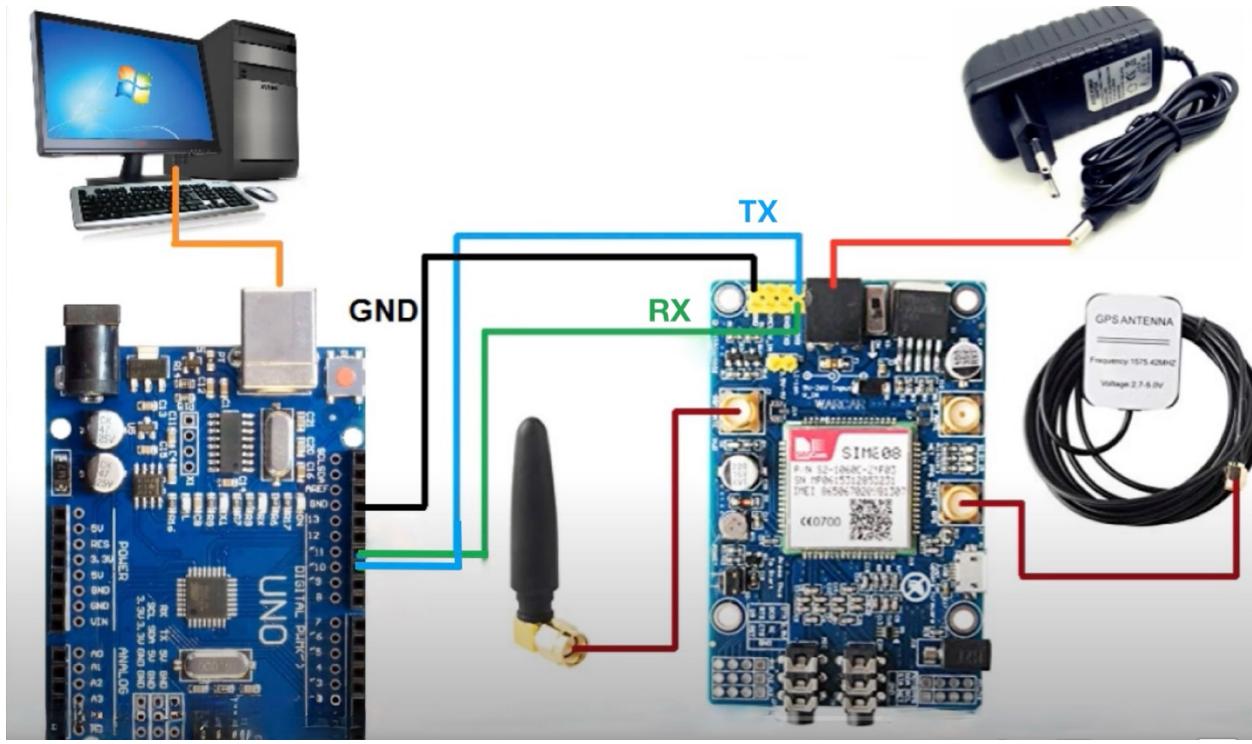
let marker = L.circleMarker([35.724688, 51.387633], {
  radius: 10,
  color: "white",
  weight: 4,
  fillColor: "#3388ff",
  fillOpacity: 1,
}).addTo(map);

let halo = L.circle([35.724688, 51.387633], {
  radius: 50,
  color: "",
  fillColor: "#38A8FF",
  fillOpacity: 0.2,
  weight: 0,
}).addTo(map);

let firstTime = true;

setInterval(() => {
  fetch("location.txt?" + new Date().getTime())
    .then((r) => r.text())
    .then((text) => {
      const [lat, lon] = text.trim().split(",");
      const latNum = parseFloat(lat);
      const lonNum = parseFloat(lon);
      if (!isNaN(latNum) && !isNaN(lonNum)) {
        const position = [latNum, lonNum];
        marker.setLatLng(position);
        halo.setLatLng(position);
        if (firstTime) {
          map.panTo(position);
          firstTime = false;
        }
        console.log("Updated to:", position);
      }
    })
    .catch((e) => console.log("location.txt not found or invalid"));
}, 1000);
</script>
</body>
</html>
```

پیوست ۴: شماتیک مدار



• مراجع ٩

SIMCom Wireless Solutions. SIM808 Hardware Design V1.00.

V. Agafonkin, "Leaflet – an open-source JavaScript library for mobile-friendly interactive maps," 2023. [Online]. Available: <https://leafletjs.com>.

OpenStreetMap Foundation, "OpenStreetMap Wiki," 2023. [Online]. Available: <https://wiki.osm.org>.

C. Liechti, "pySerial Documentation," 2023. [Online]. Available:
<https://pyserial.readthedocs.io>.

Arduino, "Arduino Language Reference," 2023. [Online]. Available:
<https://www.arduino.cc/reference/en/>.

نقشه کاربری اراضی تهران HydroGIS, ".