

**EDITOR-IN-CHIEF:**

DR. FATOS XHAFA

Universitat Politècnica de Catalunya  
Barcelona, Spain

CHECKING AND MONITORING THE DRIVING OF  
DRIVERS IN THE TRANSFORMING FLEET OPERATIONS

## IoT Final project Report



MEHDI MOOSAVIUN

SANAZ MOTIE

Dr.Reza Vahidnia

Fall 2023

# TABEL OF CONTANTS

Problem	3
Solution	4
Hardware	5
Sim800I	6
Sim800I Commands	13
UART to USB Convertor	13
ESP32	18
MPU6050	23
Sending Data to Server	28
IoT Platform	30
Conclusion	32
Future Features	33
References	34



## PROBLEM

صنعت حمل و نقل و ارسال مرسولات یکی از قطب های حیاتی اقتصادی می باشد. در این صنعت، تحویل سریع و امن مرسولات بده برای افراد و شرکت ها امری حیاتی است. همواره ارسال مرسولات و نقل و انتقال آنها یک چالش جدی در صنعت حمل و نقل محسوب شده است. در طی این عملیات، مرسولات ممکن است با خطرات مختلفی مواجه شوند که به وسیله آنها آسیب دیده و ممکن است ارزش آنها کاهش یابد یا از بین بروند. این مسئله می تواند نتایجی بسیار هزینه بر و منفی در میان خریداران و شرکت های حمل و نقل ایجاد کند. بنابراین، شناسایی دقیق عوامل و مشکلات مرتبط با آسیب رسیدگی به مرسولات بده و ابداع راه حل های موثر در این خصوص از اهمیت بالایی برخوردار است.

در ارسال مرسولات چندین عامل می تواند مؤثر باشد و آسیب رسیدگی به مرسولات بده را افزایش دهد. به عنوان مثال، عواملی مانند دسترسی نامناسب به مرسولات بده و ناهمسو با نیازهای آنها می توانند به خرابی آنها انجاماند. همچنین، تغییرات دما، رطوبت و شرایط ناهماهنگ حمل و نقل ممکن است روی مرسولات اثر بگذارد. علاوه بر این، عملیات بارگیری و تخلیه نادرست می تواند منجر به خرابی مرسولات بده شود.

پس در زمینه حمل و نقل مرسولات ما با چالش هایی روبه رو هستیم:

- آگاهی از سرعت راننده و کنترل آن
- آگاهی از موقعیت مکانی راننده
- آگاهی از وضعیت راننده اعم از اینکه راننده در حین ارسال مرسوله تصادف نکرده باشد
- آگاهی از شرایط مورد نیاز مرسوله اعم از شرایط دمایی و فشار و رطوبت
- الگوهای نامطلوب رانندگی مانند شتاب دهی شدید و ترمززدن سخت

با توجه به این چالش ها ما باید یک راه حل جامع و کامل برای مانیتور کردن وضعیت رانندگان داشته باشیم.





## SOLUTION

در دنیای پیشرفته امروزی، اینترنت اشیا به عنوان یک تکنولوژی نوآورانه و پیشرانه، نقش مهمی در فرآیندهای صنعتی و زندگی روزمره ما ایفا می‌کند. اینترنت اشیا به راه‌اندازی ارتباطات میان شیء و انتقال داده‌ها بین آنها از طریق شبکه‌های اینترنت متصل به یکدیگر اشاره دارد. اینترنت اشیا به طور عمده در بستر جمع‌آوری داده‌های آشکار (sensor data) اشیا و تبادل این داده‌ها با سرورها و سیستم‌های ذخیره‌سازی مورد استفاده قرار می‌گیرد.

یکی از حوزه‌های کاربردی بسیار مهم اینترنت اشیا در صنعت حمل و نقل و ارسال مرسولات است. به منظور بهبود کارایی و افزایش سطح امنیت در این صنعت، بررسی و مانیتور کردن رانندگان و وسایل نقلیه استفاده شده در ارسال مرسولات امری حیاتی به حساب می‌آید. برای جمع‌آوری داده‌های مربوط به رانندگان، می‌توان از سیستم‌های ردیابی GPS، حسگرهای فشار سوخت، سنسورهای اتاق کابین و موارد مشابه استفاده کرد تا اطلاعات مرتبط با راننده مثل سرعت، موقعیت جغرافیایی، تماس‌ها و سایر ویژگی‌های مربوط به راننده را به صورت لحظه‌ای ارسال کند.

داده‌ها با استفاده از سنسورهای بکارگرفته شده در خودرو و به وسیله ارتباط cellular برای پلتفرم ما ارسال میشوند و این داده‌ها توسط اپراتور مانیتور شده و در صورت تخطی از شرایط تعیین شده یا تغییر شرایط مورد نیاز مطابق دستور العمل‌ها اقدامات لازم انجام میشود.

در این گزارش پروژه، ما تلاش خواهیم کرد تا یک سامانه مبتنی بر اینترنت اشیا برای بررسی و مانیتور کردن رانندگان در ارسال مرسولات ایجاد کنیم. با استفاده از داده‌های جمع‌آوری شده و تحلیل موردی، قصد داریم تا بهبودی در عملکرد رانندگان، بهبود امنیت حمل و نقل و بهبود کیفیت خدمات مرسولات بده را در این صنعت به ارمغان بیاوریم. این گزارش شامل معرفی مسئله، طراحی و پیاده‌سازی سامانه و تحلیل داده‌ها جهت بهبود فرآیندها خواهد بود.

- پس راه حل ما برای این مشکل استفاده از ماژولی است که میتواند از طریق پیامک موقعیت مکانی و سرعت را ارسال کرده و همچنین در شرایط اضطراری با ارسال پیامک ما را آگاه کند و داده را به سرور منتقل کند تا اپراتور آن را مانیتور کند. این ماژول و نحوه ارسال دیتا و کارکرد آن در ادامه بطور مفصل توضیح داده شده است.

در ادامه به تحلیل داده‌ها، سخت افزار مورد نیاز برای این کار شامل سنسورها و عملگرها، نحوه ارسال داده از سنسور به پلتفرم و ارتباطات، تحلیل‌های آماری داده، داشبورد و مانیتورینگ داده‌ها و اقدامات لازمه در صورت تغییر شرایط لازمه می‌پردازیم.



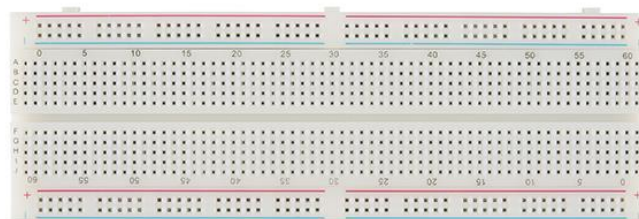


# HARDWARE

از آنجا که ویژگی اصلی این مسئله ارسال داده از ماشین های درون جاده است پس نمیتوانیم از ارتباطاتی چون BLE LORA - WIFI استفاده کنیم چونکه این تجهیزات در فاصله های کم قابل استفاده هستند و برای هدف ما که فاصله ها کیلومتری است پاسخگو نیستند. پس ما باید از یک سیستم ارتباطی دیگر استفاده کنیم، این سیستم CELLULAR نام دارد که در ادامه به طور مفصل شرح داده میشود.

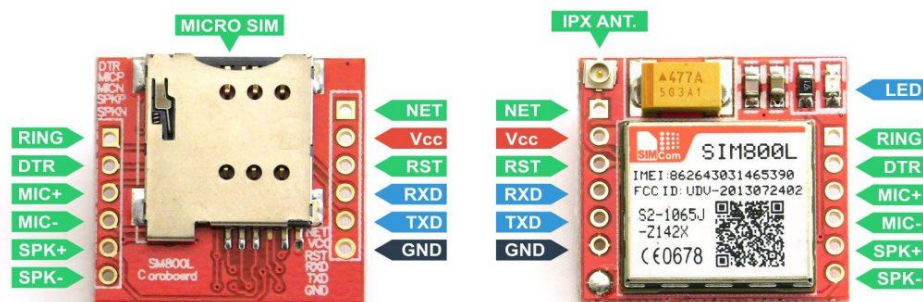
سخت افزار لازم برای انجام این پروژه شامل:

**Battery li-ion 3.7v – Adapter 12v & 2A – tp4056 – Serial to USB convertor - ESP32 – sim800l – mpu6050 - Regulator LF – PCB board – Bread board**



## • Sim800l

ماژول SIM800L یک ماژول GSM/GPRS کوچک است که توسط شرکت SIMCom طراحی و تولید می‌شود. این ماژول قابلیت برقراری ارتباطات تلفن همراه و انتقال داده‌ها را از طریق شبکه GSM را داراست. SIM800L یکی از محبوب‌ترین ماژول‌های GSM/GPRS در حوزه ساخت وسایل الکترونیکی و IoT (اینترنت اشیاء) است. از این ماژول برای ارسال و دریافت پیامک (SMS)، برقراری تماس صوتی (با استفاده از میکروفون و بلندگوی خارجی) و انتقال داده‌ها از طریق اینترنت با استفاده از GPRS استفاده می‌شود. همچنین می‌توان از آن به عنوان یک مودم GSM برای برقراری ارتباط با سرورها و سایر دستگاه‌ها استفاده کرد. ماژول SIM800L از طریق پایه‌هایی که در آن وجود دارند، به میکروکنترلرها و سایر وسایل الکترونیکی متصل می‌شود. برای کنترل ماژول و ارسال دستورات، از طریق اتصال سریال (Serial) با سرعت بیت‌های بالا اقدام می‌شود. مزیت‌های استفاده از ماژول SIM800L شامل اندازه کوچک، مصرف انرژی پایین، امکانات برقراری تماس صوتی و ارسال پیامک، قابلیت اتصال به اینترنت و سازگاری با استانداردهای GSM/GPRS است. به علاوه، SIM800L دارای آنتن داخلی است و می‌توان آن را با باتری راه‌اندازی کرد. ماژول SIM800L، یک ماژول مخابراتی از نوع سیم کارتی است. این ماژول یک اسلات سیم کارت داشته که سیم کارت از نوع mini درون آن قرار می‌گیرد. با اتصال سیم کارت، این ماژول قادر خواهد بود که همانند یک تلفن همراه، به شبکه مخابرات متصل شده و به ارسال دریافت پیامک، برقراری تماس تلفنی و اتصال به اینترنت بپردازد. ماژول SIM800L دارای دو پایه سریال، RX و TX جهت دریافت فرمان و ارسال داده است. این ماژول همچنین بر روی خود پایه‌های اتصال میکروفون و بلندگو جهت برقراری تماس تلفنی است.



## مشخصات فنی ماژول SIM800L عبارتند از:

فرکانس‌های پشتیبانی: Quad-Band 850/900/1800/1900MHz

پشتیبانی از GPRS multi-slot class 12 و اتصالات GPRS/EDGE

سیم کارت: پشتیبانی از سیم کارت ۱۰۸ ولت و ۳ ولتکانکتور آنتن: سوکت U.FL

اتصال سریال: پورت UART با سرعت تا ۱۱۵۲۰۰ بیت بر ثانیه

ورودی/خروجی صوتی: بلندگوی خارجی و میکروفون خارجی

ارسال و دریافت پیامک (SMS) با پشتیبانی از PDU و TEXT mode

ارسال و دریافت دیتا از طریق اینترنت با پشتیبانی از GPRS (با حداکثر سرعت دانلود ۸۵٫۶ کیلوبیت بر ثانیه و

سرعت آپلود ۴۲٫۸ کیلوبیت بر ثانیه)

برقراری تماس صوتی با پشتیبانی از میکروفون و بلندگوی خارجی

ارسال و دریافت دیتا با استفاده از Tone Multi-Frequency-DTMF Dual

پشتیبانی از Caller ID و ثبت شماره‌ی مخاطبین در حافظه

پشتیبانی از USSD (Unstructured Supplementary Service Data)

پشتیبانی از CSD (Circuit Switched Data)

ابعاد فیزیکی:  $3 \times 24 \times 24$

وزن: تقریباً ۳٫۵ گرم

ولتاژ کاری: ۳٫۴ تا ۴٫۴ ولت (توصیه می‌شود ولتاژ ۴ ولت برای کارکرد بهینه استفاده شود)

مصرف جریان در حالت آماده‌به‌کار: کمتر از ۲ میلی آمپر

مصرف جریان در حالت انتقال داده: کمتر از ۵۰۰ میلی آمپر

**انواع ماژول‌های سیمکارت SimCom:** این شرکت انواع ماژول‌های نسل دوم، نسل سوم و اخیراً نسل چهارم

را در دست تولید دارد. از دیگر محصولات این شرکت می‌توان به تولید ماژول‌های GPS نیز اشاره نمود. ماژول

sim800c, sim808, sim800L, sim800A از پرکاربردترین ماژول‌های مخابراتی شرکت سیم کام در ایران

هستند.



## مشخصات پایه های ماژول Sim800I

اتصال آنتن و لحیم کردن آن روی برد	NET
تغذیه بین 3.4-4.4 ولت	VCC
پین ریست ماژول برای فعال سازی باید LOW شود.	RST
گیرنده ارتباط سریال	Rx
فرستنده ارتباط سریال	Tx
پین منفی تغذیه - این پین باید به GND میکروکنترلر یا رابط سریال نیز متصل شود.	GND
بیانگر برقرای تماس. از این پایه میتوان بعنوان ایجاد وقفه خارجی نیز استفاده کرد. در حالت معمولی وضعیت این پایه HIGH است و در صورتی که با ماژول تماس گرفته شود، وضعیت آن به مدت 120 میلی ثانیه LOW خواهد شد.	RING
فعال یا غیر فعال کردن حالت Sleep. اگر HIGH شود ماژول در حالت Sleep قرار گرفته و ارتباط سریال غیر فعال خواهد شد.	DTR
پین ورودی میکروفون	MIC±
پین اتصال بلندگو	SPK±

## تغذیه ماژول Sim800I

تغذیه اولین حرف را در راه اندازی این ماژول میزند. بطوری که اگر تغذیه مناسب نباشد ماژول مدام ریست شده و به شبکه متصل نمی شود. برای اتصال موفقیت آمیز ماژول های GSM به شبکه باید به تغذیه آن دقت لازم را داشت. منبع مورد استفاده باید قابلیت جریان دهی ۳ آمپر را داشته و ولتاژ مورد نیاز آن هم بین ۳.۴ تا ۴.۴ ولت باشد البته در پروژه ما در ابتدا این ماژول روشن نمیشد و مدارم ریست میشد که با آزمایش و خطا متوجه شدیم در ولتاژ ۴.۵ ولت اتصال آن

موفقیت آمیز است و راه اندازی شد.

حالت کاری	فرکانس	جریان مصرفی
Power down	-	60uA
Sleep mode	-	1mA
Stand by	-	18mA
Call	GSM850	199mA
	EGSM900	216mA
	DCS1800	146mA
	PCS1900	131mA
GPRS	-	453mA
Transmission burst	-	2A



برای تغذیه ماژول می‌توانید از باتری های Li-ion که رنج ولتاژ بین ۳.۷ تا ۴.۲ ولت هست استفاده کنید

در مورد پروژه ما چونکه باتری تخلیه میشد و نیاز به شارژر داشت در ابتدا از آداپتور و رگولاتور

استفاده کردیم و مدار را بستیم و کد ها را بر روی esp32 کامپایل کردیم و در نهایت چون

اساسا این پروژه ها باید بصورت ریموت باشند از باتری استفاده کردیم که در ادامه توضیحات

بیشتری داده میشود.

یکی دیگر از گزینه های موجود برای تغذیه ماژول، استفاده از آداپتور می‌باشد. برای کاهش ولتاژ در حد مورد نیاز باید

از مبدل های DC-DC نوع باک میشه استفاده کرد.

در این مورد ما از آداپتور ۱۲ ولت و ۲ آمپر استفاده

کردیم و آن را به رگولاتور متصل کردیم تا خروجی

مورد نظرممان را به ما بدهد.



برروی برد راه انداز ماژول SIM800L، یک LED که بیانگر وضعیت ماژول می‌باشد وجود دارد. نحوه چشمک زدن

آن بیانگر حالت های مختلف هست. اگر ماژول به شبکه وصل نشده باشه LED سریع چشمک می‌زنند. در صورتی که

شبکه را پیدا کرده و به آن کانکت شود، LED هر ۳ ثانیه یک بار چشمک خواهد زد. در زیر وضعیت نمایش LED

شبکه ماژول را مشاهده میکنید.

از جمله نکات مهمی که در طول راه اندازی این ماژول به آن برخورد کردیم عدم اتصال ماژول به شبکه بود که

راه حل های موجود برای دیباگ کردن آن به شرح زیر است:

در اتصال سیم های تغذیه به ماژول از سیم با ضخامت مناسب و قوی استفاده کنید. که البته در پروژه ما با سیم جامپر

استفاده کردیم و به نتیجه مطلوب رسیدیم. در صورتی که مشاهده شد LED روی برد سریع چشمک می‌زنند، بعدش

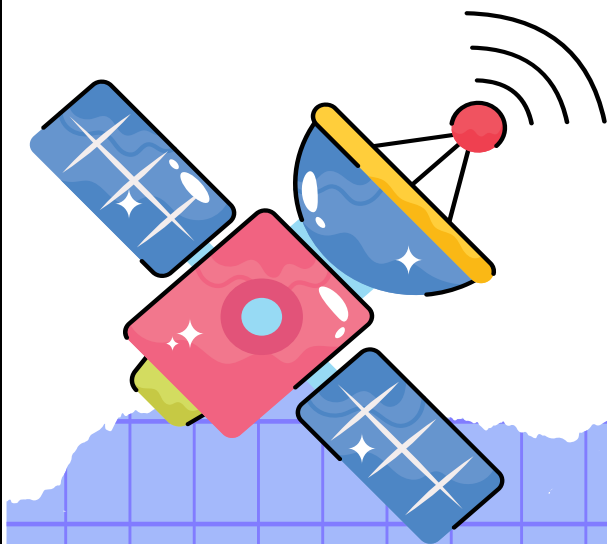
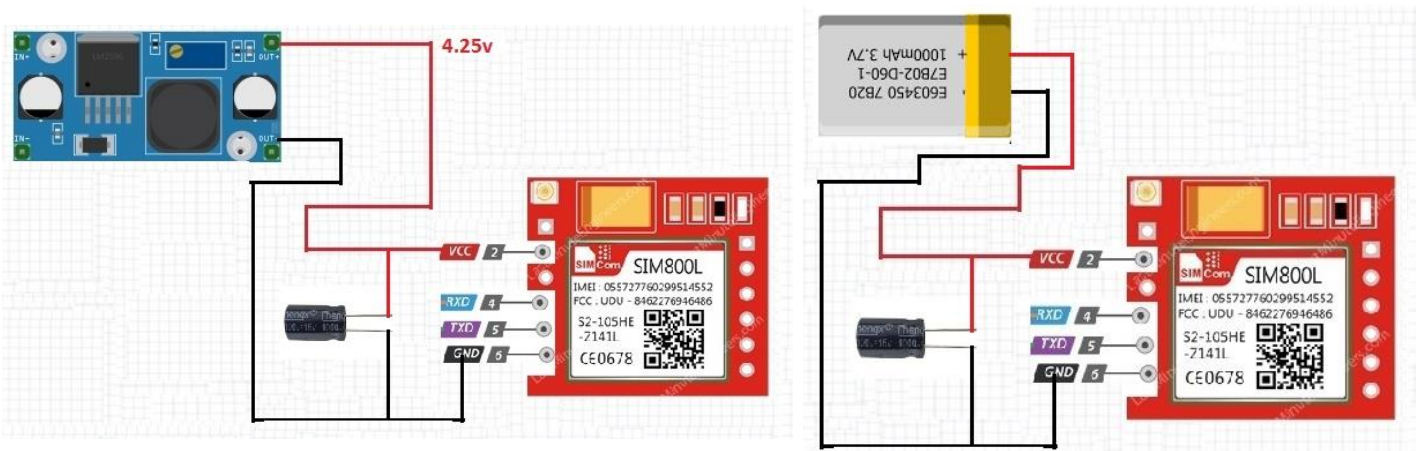
چند ثانیه خاموش شد و مجددا شروع به چشمک زدن کرد. ماژول دارد ریست میشود. پس اتصالات و حتما تغذیه را

بررسی کنید. برای کسب نتیجه بهتر از آنتن خارجی با گین بالاتر استفاده کنید. منبع مورد استفاده شده قابلیت جریان

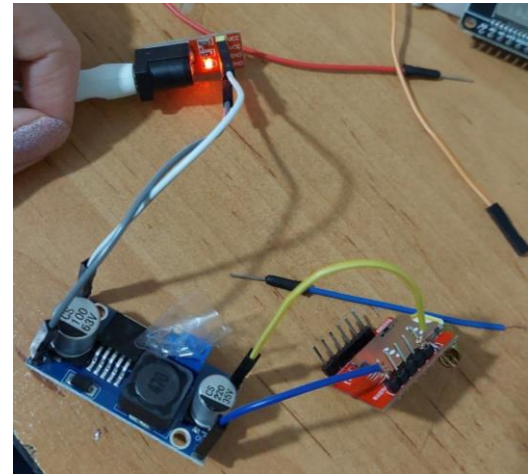
دهی حداقل ۲-۳ آمپر را داشته باشد. و تا جای ممکن از تماس دست خود با خود ماژول SIM800L جدا خودداری کنید. این ماژول ها در برابر الکتریسته ساکن خیلی حساس بوده و امکان سوختن و یا آسیب به آنها خیلی زیاد است. از غیر فعال بودن پین کد سیم کارت خود مطمئن شوید. در صورتی که فعال بود میتوانید با گذاشتن سیم کارت روی یک گوشی، از بخش تنظیمات اون رو غیر فعال کنید.

از مشکلاتی که ما در حین راه اندازی ماژول با آن مواجه بودیم متصل نشدن آن به شبکه بود که در ابتدا ما سیم های اتصال را تغییر دادیم و از سیم های مسی ضخیم تر استفاده کردیم که مجددا اتصال برقرار نشد در تلاش بعدی با آزمایش ولتاژ داده شده به ماژول به بررسی اتصال آن پرداختیم و نهایتا نکته مهمی که به آن برخوردیم موازی سازی یک خازن با ماژول بود که پس از آن ماژول به شبکه متصل شد. این خازن ۱۰۰۰ میکروفاراد و با ولتاژ بالای ۱۰ ولت بود که باید مورد استفاده قرار می گرفت.

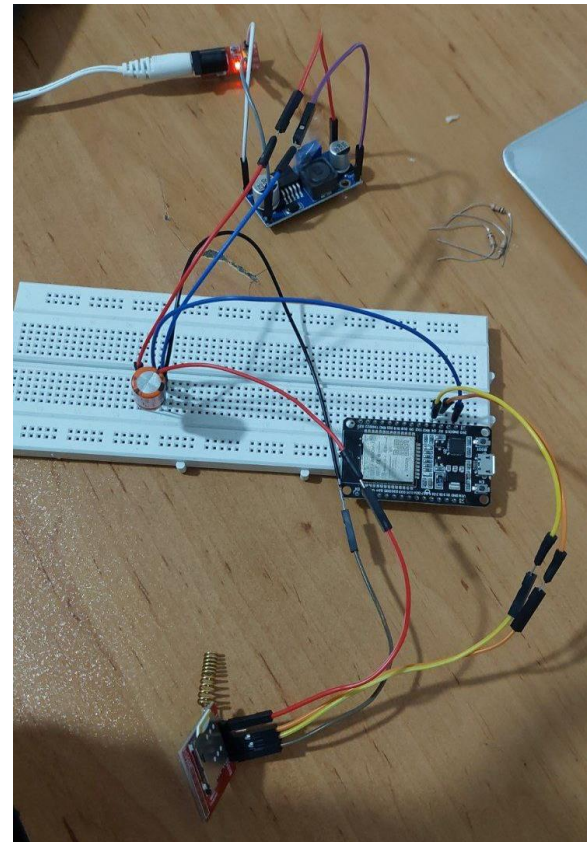
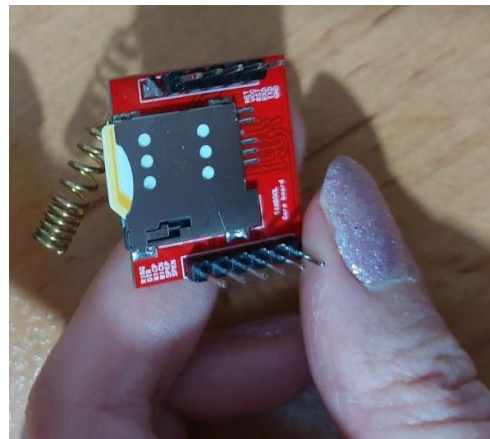
شماتیک اتصالات آن بصورت زیر است:



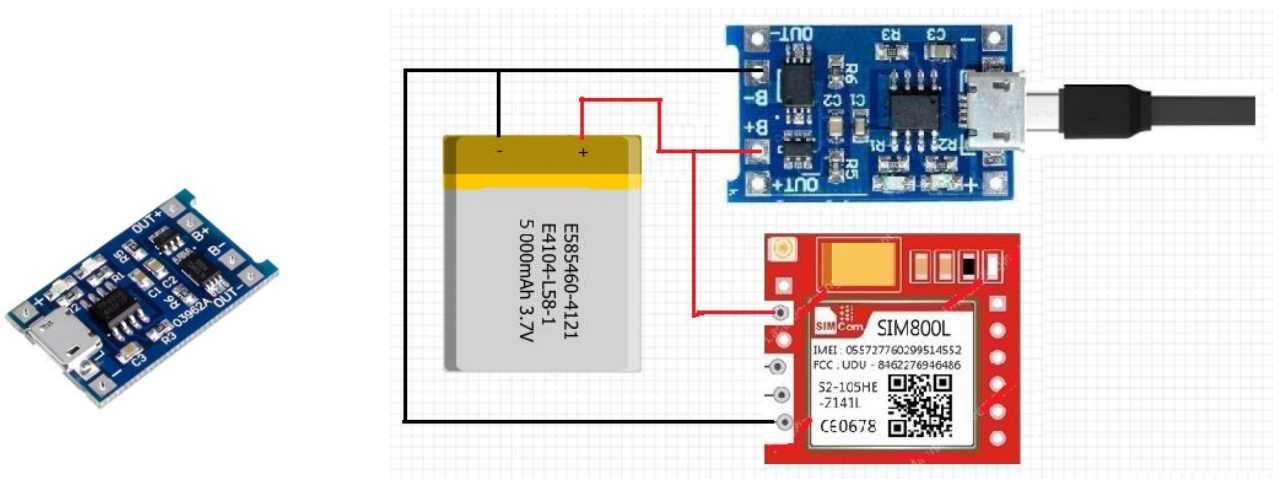
در ادامه تصاویری از آزمایش و خطاهای انجام شده برای راه اندازی ماژول آورده شده است که در ادامه شرح داده میشود.



با توجه به تصاویر سیم بندی بالا در ابتدا ما با استفاده از آداپتور و رگولاتور و بدون موازی سازی خازن سعی در راه اندازی ماژول داشتیم که راه اندازی نشد، سپس با چک کردن نحوه قرارگیری سیمکارت را چک کردیم که درست بود و در نهایت با مطالعات بیشتر با موازی سازی خازن با ماژول ، ماژول ما مطابق شکل های زیر راه اندازی شد.



در ادامه به سراغ پرتابل سازی ماژول خود رفتیم. با توجه به آنکه این ماژول برای ردیابی وضعیت ماشین در جاده است پس نیاز است بصورت پرتابل در ماشین قرار گیرد زیرا در ماشین ما به پریز برق دسترسی نداریم پس ما ابتدا مطابق مدار زیر به سراغ پرتابل سازی با شارژ همزمان باتری رفتیم که میتوان شارژر را از باتری جدا کرد و یک مدار مجتمع تقریباً کوچک را برای ردیابی، ارسال داده و ارسال پیامک در ماشین قرار داد بدون آنکه نیازی به اتصال دائمی به برق داشته باشد.



یکی از مهم ترین چالش ها در راه اندازی ماژول SIM800L به کمک باتری، نحوه شارژ باتری است. برای شارژ باتری های لیتیم یونی و لیتیم پلیمری، می بایست روال خاصی را در پیش گرفته و به طراحی مدار بپردازید. این مورد می تواند پیچیدگی های خاص خود را داشته باشد. این در حالیست که به کمک ماژول شارژر باتری tp4056، می توانید به سادگی و بدون نیاز به هیچ ابزار جانبی، باتری های لیتیم یون و لیتیم پلیمر تک سلول را شارژ کنید. به کمک این ماژول می توانید باتری خود را از طریق شارژرهای تلفن همراه شارژ کنید. همچنین بر روی این ماژول چراغ نشانگر جهت اعلام شارژ شدن باتری به کاربر تعبیه شده است. در ادامه به کمک شماتیک زیر می توانید باتری متصل به ماژول را بدون جداسازی باتری از مدار، شارژ کنید. ماژول شارژر باتری پس از شارژ کامل و قرار گرفتن ولتاژ باتری بر روی ۴/۲۵ ولت، جریان الکتریکی را قطع می کند.





## • ارسال دستورات به Sim800l

در بخش دوم و پس از آشنایی با ماژول Sim800l به سراغ نحوه ارسال دستورات به Sim800l میرویم.

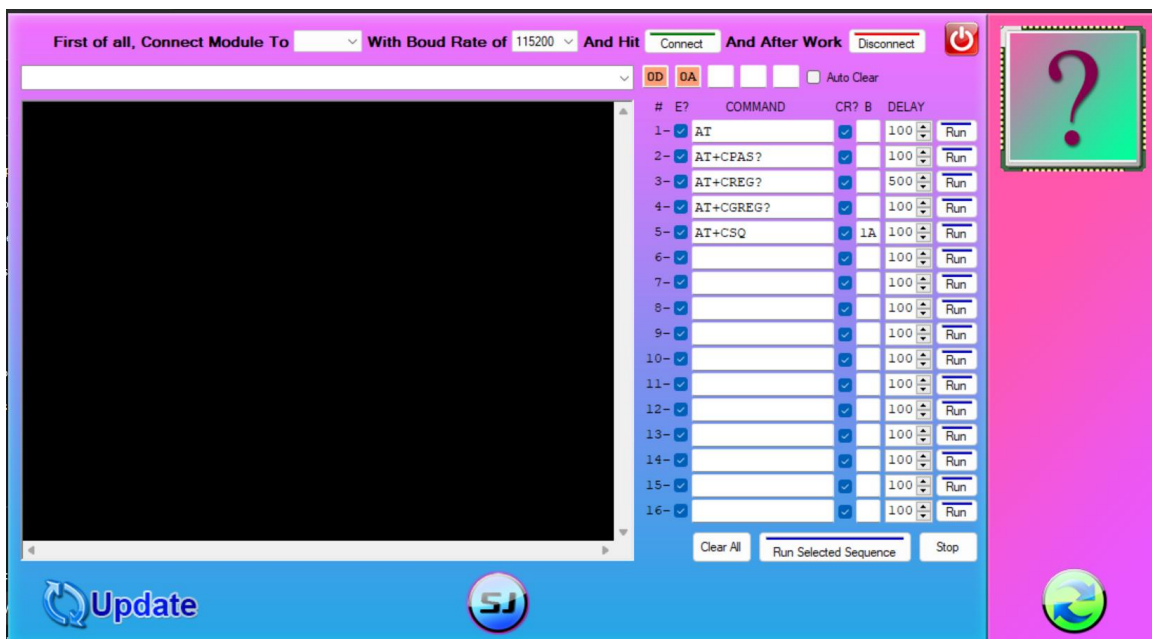
در ابتدا نکته ای که مهم است ویژگی های مورد نیاز ما برای این پروژه است، در این پروژه ما نیاز داریم که با ارسال پیامک به ماژول خود، ماژول به ما سرعت و موقعیت مکانی را بدهد. همچنین در صورت تشخیص تصادف با شماره ای که در کد نوشته شده است تماس بگیرد و در صورت عدم تماس، پیامک بدهد.

نکته بعدی ارسال داده ها به سرور ما است و که بتوانیم این داده ها به سرور ارسال کنیم و در پلتفرم خود آن را مانیتور کنیم. پس در ادامه به سراغ آشنایی با انواع دستورات به ماژول و نحوه ارسال آن میپردازیم.

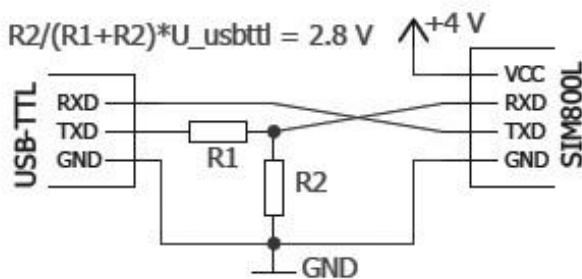
برای ارسال دستورات ما به دو صورت میتوانیم این کار را انجام دهیم، اول با استفاده از ESP32 و شیوه دوم بصورت مستقیم و با استفاده از Serial to USB convertor میشود دستورات را به ماژول ارسال کرد.

### ۱- Serial to USB convertor

یک روش برای ارسال دستور از کامپیوتر به ماژول از طریق Serial to USB convertor است. ما برای این پروژه در ابتدا به مشکل خوردیم و مشکل اساسی دوم پس از عدم اتصال ماژول به شبکه، عدم دریافت جواب از سمت ماژول بود. برای حل این مشکل ما دو راه در پیش رو داشتیم اول استفاده از Serial to USB convertor و دوم استفاده از ESP32 است. در ابتدا به سراغ Serial to USB convertor رفتیم. در این روش از یک نرم افزار ارتباط سریال sj-ATCTool مخصوص ماژول های GSM استفاده کرده شد که صفحه اصلی آن به شکل زیر است.

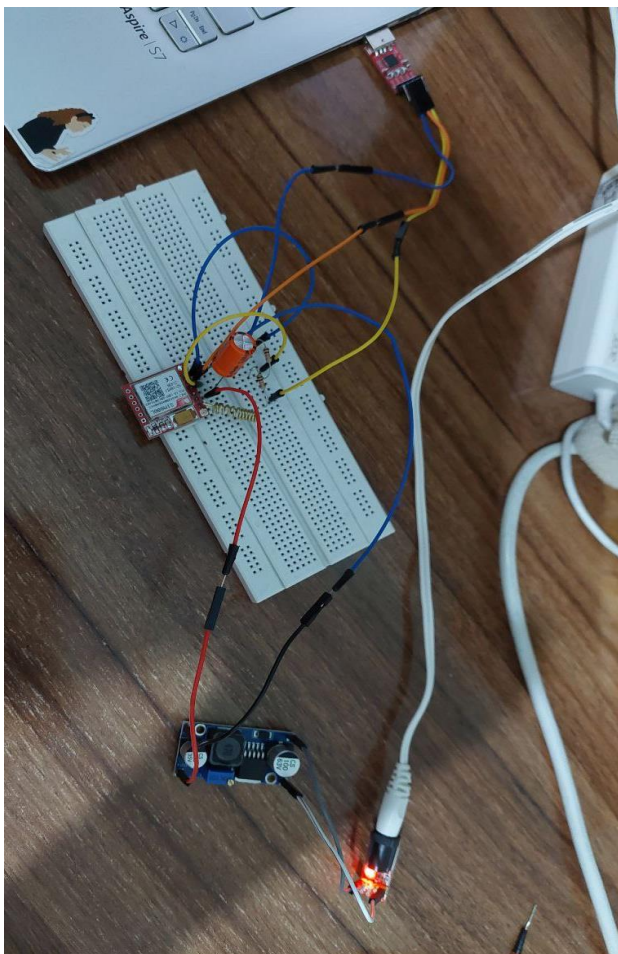


در ابتدا سیم بندی و اتصالات مدار را به شکل زیر بستیم و سپس از دستورات استفاده کردیم.



- مطابق شکل های رو به رو ابتدا برای اتصال ماژول به مبدل از نقشه روبه رو استفاده میکنیم که پین ارسال دیتا ماژول را به دریافت مبدل و پین دریافت دیتا ماژول را به ارسال دیتا مبدل اتصال میدهم. نکته حائز اهمیت در این بخش اتصال زمین های هر دو ماژول به یکدیگر است زیرا در غیر اینصورت اتصال برقرار نمیشود.

مطابق شکل روبه رو اتصالات را به شیوه روبه رو میبندیم که نهایتا در شکل زیر اتصالات ما آورده شده است. پس از اتصالات به سراغ ارسال داده از طریق مبدل به ماژول میرویم که در ادامه بطور مفصل به توضیح آن پرداخته شده است.



برای ارسال دستورات به ماژول باید از AT command استفاده کنیم.

با استفاده از دستور AT+CCALR? و پاسخ ارسالی از طرف ماژول میتوانیم متوجه بشویم که ماژول آماده هست یا

خیر. ماژول در پاسخ به این دستور اگر همه چیز اوکی باشه و آماده phone call باشه، ۱ ارسال خواهد کرد.

```
AT+CCALR?
```

```
+CCALR: 1
```

```
OK
```

اول دستور AT را میفرستیم و مطمئن میشویم که ماژول OK رو در پاسخ به ما میدهد. این را برای مطمئن شدن از

برقرای اتصال سریال انجام میدیم. سایر مراحل نیز در زیر آورده شده است.

### اطمینان از برقرای ارتباط سریال:

```
AT
```

```
OK
```

### اطمینان از متصل شدن به شبکه:

```
AT+CSQ?
```

```
+CSQ: 31,0
```

### تنظیم ماژول در مد کاری Text:

```
AT+CMGF=1
```

```
OK
```

### وارد کردن شماره مقصد مورد نظر:

```
AT+CMGS="09219656927"
```

```
یا
```

```
AT+CMGS="+989219656927"
```

```
>
```

بعد از تایپ کردن متن مورد نظر، حتما باید قبل از زدن دکمه اینتر کیبورد، عبارت ctrl+z یا کد هگز (1A) معروف

هم بهمراش ارسال بشود تا وقتی که عبارت ctrl+z ارسال نشود، هر چی ارسال بشود به خط بعدی پیامک مورد ارسال

میرود.



```
AT+CMGS="09219656927"
```

```
>https://blog.microele.com
>SIM800L Tutorial - Part 2
>Visit our new post on Microelecom Blog
>
```



```
+CMGS: 76
```

```
OK
```



در این بخش تنظیمات را مشابه حالت قبل بررسی و اعمال خواهیم کرد. هدف این بخش، دریافت پیامک می‌باشد. اول دستور AT را می‌فرستیم و مطمئن میشیم که ماژول OK رو در پاسخ به ما میده. این را برای مطمئن شدن از برقرای اتصال سریال انجام میدیم.

### اطمینان از برقرای ارتباط سریال:

```
AT
```

```
OK
```

### اطمینان از متصل شدن به شبکه:

```
AT+CSQ?
```

```
+CSQ: 31,0
```

### خواندن پیامک از روی حافظه سیم کارت:

با استفاده از دستور زیر و مشخص کردن آدرس پیامک در خانه حافظه سیم کارت، میتوانید پیامک ذخیره شده در آن را بخوانید. بجای XX میتوانید آدرس مد نظر را وارد کنید.

```
AT+CMGR=xx
```

```
AT+CMGR=1
```

```
+CMGR: "REC UNREAD","+989219656927","", "24/1/25,20:38:22+14"
light1on
```

```
OK
```

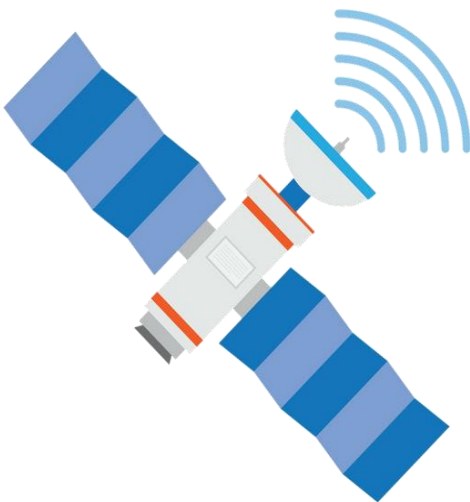
### تنظیم ماژول در مد کاری Text:

```
AT+CMGF=1
```

```
OK
```

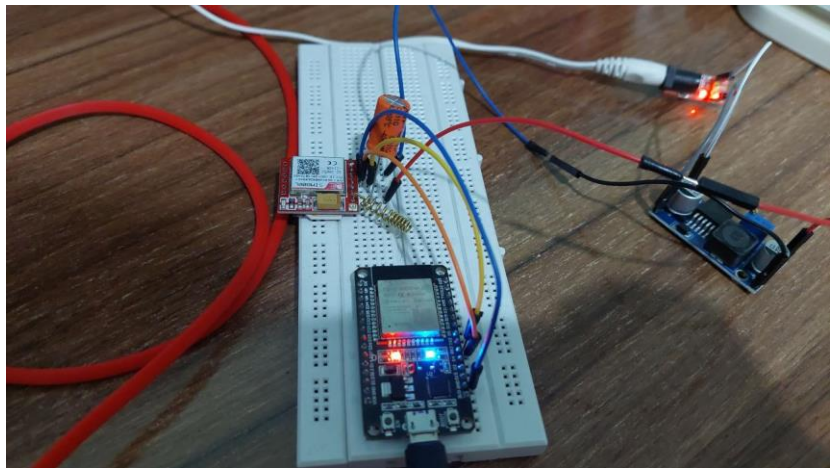
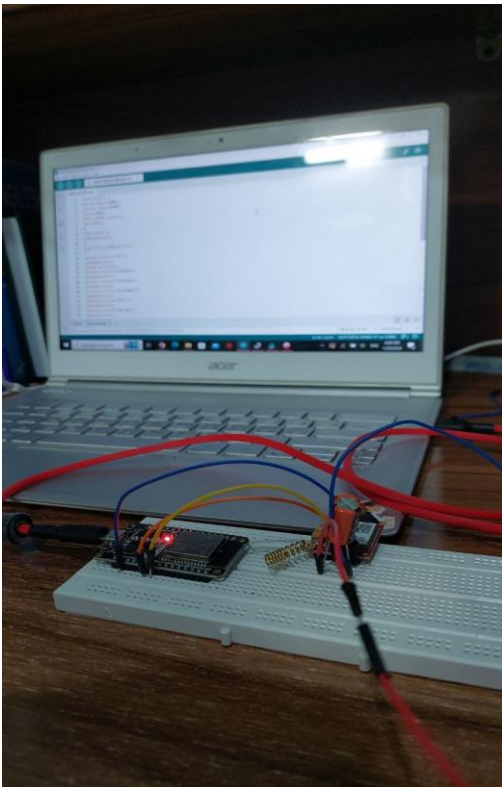
در پاسخ دریافتی از ماژول میتوان به متن پیام، تاریخ، شماره فرستنده پیام دسترسی داشت. عبارت "light1on" متن این پیامک می‌باشد. از همین روش میتوان برای خواندن سایر پیامک های موجود در حافظه سیم کارت استفاده کرد. عدد آدرس پیامک در حافظه بین ۱۵ تا ۵۰ میتواند باشد.

- با این روش پروژه ما به نتیجه نرسید و پس از ارسال دستور به ماژول، ماژول پاسخی به ما نمیداد و پس از تلاش های زیاد و مطالعه بسیار مجدداً نیز به نتیجه نرسیدیم و به سراغ روش دوم یعنی استفاده از ESP32 رفتیم. در مورد دیباگ کردن آن تلاش های بسیاری انجام شد اما متأسفانه به نتیجه مطلوب نرسیدیم. چند مورد را برای رفع مشکل امتحان کردیم از جمله مشکلات ارتباط: شامل عدم برقراری ارتباط ماژول با میکروکنترلر یا دستگاه میزبان، نویزهای الکترومغناطیسی، اختلالات در سیگنال UART و اشتباهات در اتصال کابل ها می شود. مشکلات در شناسایی SIM کارت: شامل عدم تشخیص SIM کارت توسط ماژول، اطلاعات نادرست SIM کارت (مانند PIN) و مشکلات در قفل شبکه SIM کارت است. مشکلات شبکه: این مشکلات شامل ضعف سیگنال شبکه، نویزهای محیطی، تعارض بین باندهای فرکانسی، اتصال به شبکه ناموفق و اختلالات در ثبت SIM کارت در شبکه می شود. خطاهای دستورات AT: کاربران ممکن است با خطاهای مربوط به دستورات AT روبرو شوند. این خطاها ممکن است به دلیل استفاده اشتباه از دستورات، پارامترهای نادرست، عدم پشتیبانی از برخی دستورات در نسخه ماژول و خطاهای نرم افزاری باشد. مشکلات تغذیه برق: این شامل مصرف بالای باتری، نوسانات و نقص در تامین برق، ضعف در مدار تغذیه و مشکلات مربوط به تغذیه است. مشکلات نرم افزاری: این شامل نصب نادرست برنامه، خطاهای برنامه نویسی، ناسازگاری با سیستم عامل و مشکلات فریمور ماژول می باشد. اما با بررسی این مشکلات باز هم به نتیجه ای نرسیدیم و بنابراین به سراغ روش بعدی که استفاده از میکروکنترلر بود رفتیم.



## ۲- ESP32

همانگونه که گفته شد پس از عدم دریافت نتیجه مطلوب با استفاده از مبدل آنالوگ به دیجیتال به سراغ استفاده از میکروکنترلر ها رفتیم. در این روش میکروکنترلر ما پل ارتباطی بین ماژول و کامپیوتر برای ارسال دستورات به ماژول است. کد های این بخش بصورت جداگانه خدمت شما ارسال میشود اما در یک توضیح کلی در ابتدا ما از طریق اتصال میکروکنترلر خود به ماژول همانگونه که آن را به مبدل متصل کرده بودیم یعنی پین ارسال دیتای ماژول را به پین دریافت دیتا میکروکنترلر و پین دریافت را به پین ارسال متصل کرده و زمین ها را به یکدیگر متصل میکنیم و سپس از طریق کابل میکروکنترلر خود را به کامپیوتر متصل میکنیم و از Arduino IDE کد ها را نوشته و اتصال را برقرار میکنیم. نهایتا از قسمت Tools وارد بخش Terminal میشویم و کد های مربوطه که در بالا آورده شده بود را در این بخش مشاهده میکنیم. تصاویر زیر بخشی از این روند را به نمایش گذاشته است که پس از آن به شرح مفصل آن میپردازیم.



مطابق شکل های صفحه قبل ما در ابتدا اتصالات مربوطه را انجام میدهیم به این صورت که ماژول خود را به میکروکنترلر خود متصل میکنیم و تغذیه را نیز از یک آداپتور ۱۲ ولت و یک رگولاتور که ولتاژ آن را به ولتاژ مورد نیاز ما تبدیل میکند میگیریم. سپس AT Command را با ESP32 چک میکنیم که کد آن در زیر آورده شده است.

```

#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(4, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()
{
    //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
    Serial.begin(9600);

    //Begin serial communication with Arduino and SIM800L
    mySerial.begin(9600);
    Serial.print("Initializing...");
    delay(1000);
    mySerial.println("AT"); //Once the handshake test is successful, it will back to
    OK
    updateSerial();
    mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is
    the best
    updateSerial();
    mySerial.println("AT+CCID"); //Read SIM information to confirm whether the SIM
    is plugged
    updateSerial();
    mySerial.println("AT+CREG?"); //Check whether it has registered in the network

}

void loop()
{
    updateSerial();
}

void updateSerial()
{
    Serial.print("update");
    delay(500);
    while (Serial.available())
    {
        mySerial.write(Serial.read()); //Forward what Serial received to Software
        Serial Port
    }
    while(mySerial.available())
    {
        Serial.write(mySerial.read()); //Forward what Software Serial received to
        Serial Port
    }
}

```

در ادامه به بررسی چند تابع در کد پروژه و توضیح آنها میپردازیم.

```

116 void parseData(String buff)
117 {
118   Serial.println(buff);
119
120   unsigned int len, index;
121   index = buff.indexOf("\r");
122   buff.remove(0, index+2);
123   buff.trim();
124
125   if(buff != "OK"){
126     index = buff.indexOf(":");
127     String cmd = buff.substring(0, index);
128     cmd.trim();
129
130     buff.remove(0, index+2);
131
132     if(cmd == "+CMTI"){
133       index = buff.indexOf(",");
134       String temp = buff.substring(index+1, buff.length());
135       temp = "AT+CMGR=" + temp + "\r";
136       SIM800.println(temp);
137     }
138     else if(cmd == "+CMGR")
139     {
140       extractSms(buff);
141     }
142   }
143 }
  
```

Output

Sketch uses 275781 bytes (21%) of program storage space. Maximum is 1310720 bytes.

Ln 158, Col 1 DOIT ESP32 DEVKIT V1 on COM5 [not connected]

```

132 if(cmd == "+CMTI"){
133   index = buff.indexOf(",");
134   String temp = buff.substring(index+1, buff.length());
135   temp = "AT+CMGR=" + temp + "\r";
136   SIM800.println(temp);
137 }
138 else if(cmd == "+CMGR")
139 {
140   extractSms(buff);
141   Serial.println("Sender Number: "+senderNumber);
142   Serial.println("PHONE: "+PHONE);
143   if(senderNumber == PHONE){
144     if(msg == "get location"){
145       sendLocation();
146     }
147   }
148   SIM800.println("AT+CMGD=1,4"); //delete saved SMS
149   delay(1000);
150   smsStatus = "";
151   senderNumber="";
152   receivedDate="";
153   msg="";
154 }
155 ///////////////////////////////////////////////////
156 }
  
```

Output

Sketch uses 275781 bytes (21%) of program storage space. Maximum is 1310720 bytes.

Ln 158, Col 1 DOIT ESP32 DEVKIT V1 on COM5 [not connected]

تابع بالا تابع `parsedata` نام دارد که در کد ما پیامکی که برای ماژول خود ارسال کردیم را بررسی میکند و کلمات آن را از هم جدا میکند، اگر پیامک از همان شماره تلفنی باشد که ما مشخص کردیم و پیامک ما برابر `Send location` بود تابع `send location` را صدا میکند که آن تابع لوکیشن را برای ما پیامک میکند.

```

void sendLocation()
{
    Serial.println ("Sending sms");
    delay(1000);
    response = SIM800_send("ATH");
    delay (1000);
    response = SIM800_send("ATE0");
    delay (1000);

    response = ""; Latitude=""; Longitude="";

    SIM800.println("AT+CIPGSMLOC=1,1");
    delay(5000); //Request for location data

    while (SIM800.available())
    {
        char letter = SIM800.read();
        response = response + String(letter);
    }

    Serial.print("Result Obtained as:");    Serial.print(response);
    Serial.println("*****");

    prepare_message();
    delay(1000);

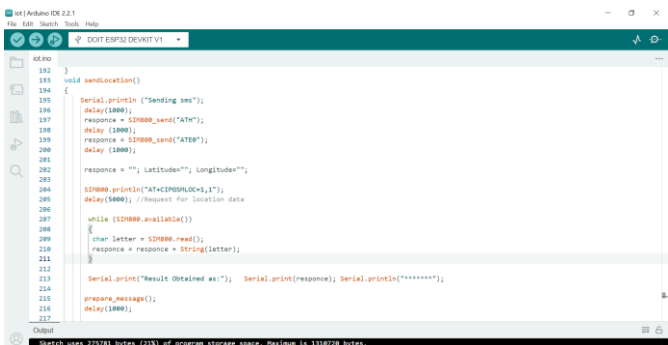
    SIM800.println("AT+CMGF=1"); //Set the module in SMS mode
    delay(1000);

    SIM800.println("AT+CMGS=\"+989219656927\""); //Send SMS to this number
    delay(1000);

    SIM800.println(Link);
    delay(1000);

    SIM800.println((char)26);
    delay(1000);
}

```



تابع بعدی ما که در بالا آورده شده است تابع `send location` است که وظیفه آن این است که ما در کد به آن `AT Command` هایی که در بالاتر تعریف شد را به آن می‌دهیم و آن به ما طول و عرض جغرافیایی را برمیگرداند و طول و عرض را به لینک گوگل مپ چسبانده و آن لینک را برای ما پیامک میکند.

تابع بعدی ما که در کد از آن استفاده شده است تابع `prepare message` است که کد آن در زیر آورده شده است.



```

72
73
74 void prepare_message()
75 {
76   int first_comma = response.indexOf(',');
77   int second_comma = response.indexOf(',', first_comma+1);
78   int third_comma = response.indexOf(',', second_comma+1);
79
80   for(int i=first_comma+1; i<second_comma; i++) //Values form 1st comma to 2nd comma is Longitude
81   {
82     Longitude = Longitude + response.charAt(i);
83   }
84   for(int i=second_comma+1; i<third_comma; i++) //Values form 2nd comma to 3rd comma is Latitude
85   {
86     Latitude = Latitude + response.charAt(i);
87   }
88   Serial.println(Latitude); Serial.println(Longitude);
89
90   Link = Link + Latitude + "," + Longitude; //Update the Link with latitude and Logitude values
91
92   Serial.println(Link);
93 }
94
95 String incoming = "";
96
97 void loop()

```

Output

Sketch uses 275781 bytes (21%) of program storage space. Maximum is 1310720 bytes.

Ln 73, Col 1 DOIT ESP32 DEVKIT V1 on COM5 [not connected]

اطلاعاتی که `Sim800l` به ما می‌دهد بین دو کاما قرار دارد، این تابع محل این کاما ها پیدا میکند و بین کامای اول تا کامای دوم `longitude` است و بین کامای دوم و سوم `latitude` قرار دارد. تابع ما مشخصات آنها را جدا میکند و به متغیر لینک اضافه کرده تا در نهایت به لینک گوگل مپ تبدیل شود.

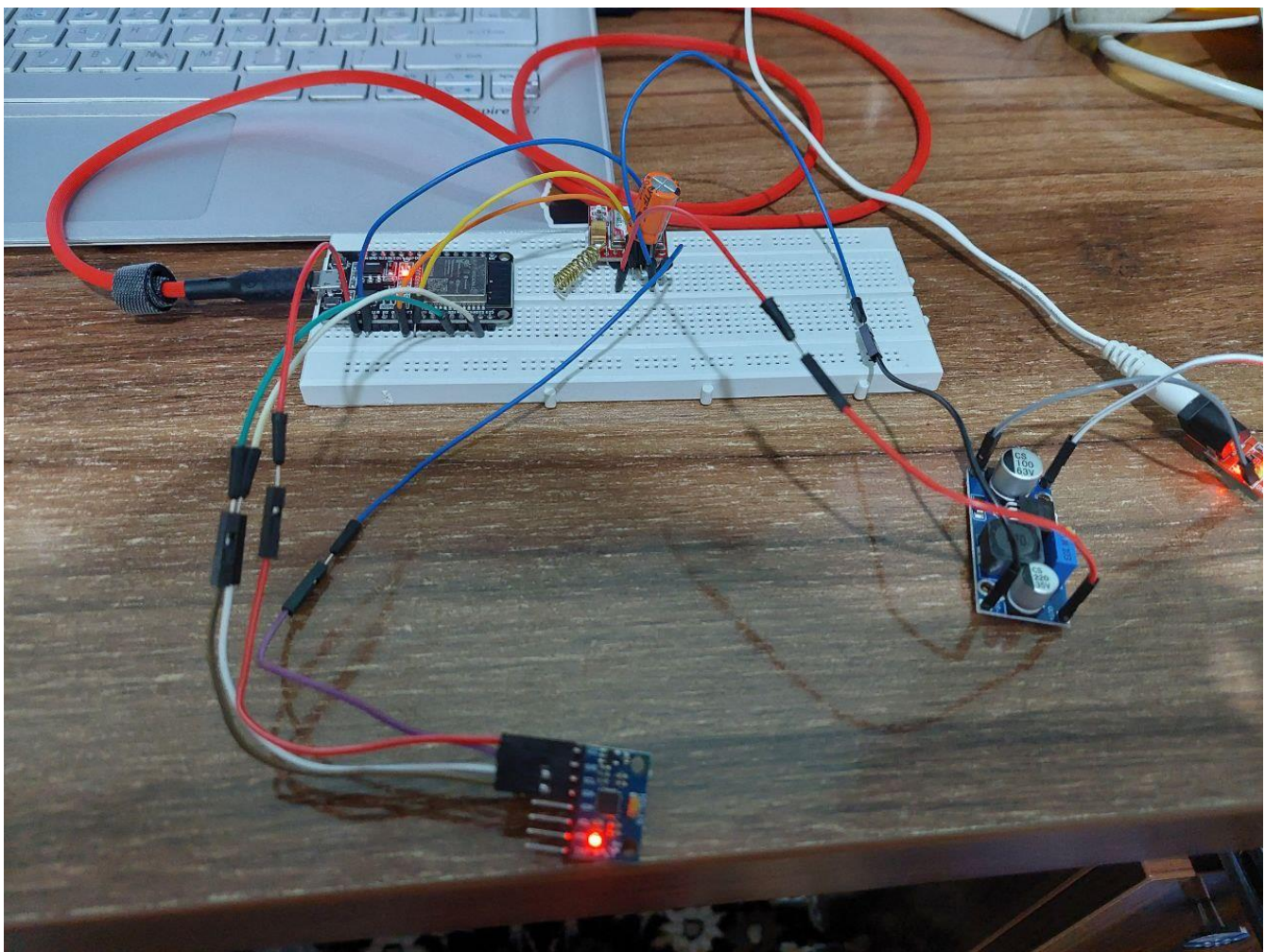
در یک توضیح کلی همانند بخش قبل که `AT Commands` را توضیح دادیم از کدهایی مشابه آنها در کد ما استفاده شده است. در یک توضیح خلاصه نوشتاری هنگامی که ما پیامی با مضمون دریافت لوکیشن به شماره که سیم کارت آن را در مازول قرار دادیم ارسال میکنیم و آن لوکیشن فعلی خودرو را برای ما میفرستد و نهایتاً یک ویژگی آن است که در صورت افزایش شتاب یا کاهش شتاب ناگهانی خودرو مازول یک پیامک خطر به شماره ای که در کد به آن دادیم ارسال میکند. در ادامه به موضوعاتی چون ارسال داده به سرور و مانیتور کردن آن نیز همچنین ارسال داده های سنسور شتاب سنج نیز میپردازیم.





## MPU6050

تا به اینجا در مورد sim800l و نحوه اتصالات آن و همچنین نحوه ارسال و دریافت پیامک و توضیح مختصری از کد های مربوط به آن داده شد. در این بخش به قسمت سنسور ما یعنی سنسور شتاب mpu6050 پرداخته میشود. کاربرد سنسور شتاب سنج برای پیش بینی تصادفات خودرو در پروژه ما استفاده میشود. برای اتصالات آن ما این سنسور را به میکروکنترلر خود با سیم بندی مطابق شکل زیر متصل میکنیم. هنگامی که شتاب ما دچار تغییرات ناگهانی شد آنگاه ماژول یک پیامک به شماره ای که در کد به آن داده شد با مضمون خطر تصادف به ما ارسال میکند.



در شکل بالا اتصالات نهایی ما برای پروژه کنترل و مانیتورینگ ناوگان رانندگان را مشاهده میکنید، همانگونه که توضیح داده شد نحوه ارسال داده ما از طریق ماژول sim800l است که از طریق کد نوشته شده در صورت ارسال پیامک در جواب لوکیشن و سرعت را برای ما میفرستد همچنین در صورت تغییر ناگهانی شتاب یا زاویه که از طریق سنسور شتاب

سنج ما تشخیص داده میشود و همچنین بیانگر تصادف است یک پیامک با مضمون خطر برای ما میفرستد. کدهای مربوطه را از طریق میکروکنترلر خود به سنسور و ماژول خود میدهیم.

در تصاویر بعدی کدهای تست سنسور mpu6050 را آورديم.

```

1 // Basic demo for accelerometer readings from Adafruit MPU6050
2
3 #include <Adafruit_MPU6050.h>
4 #include <Adafruit_Sensor.h>
5 #include <Wire.h>
6
7 Adafruit_MPU6050 mpu;
8
9 void setup(void) {
10   Serial.begin(115200);
11   while (!Serial)
12     delay(10); // will pause Zero, Leonardo, etc until serial console opens
13
14   Serial.println("Adafruit MPU6050 test!");
15 }
16
17 void loop() {
18   // Rotation
19   Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s
20   Temperature: 12.53 degC
21
22   // Acceleration
23   Acceleration X: 1.22, Y: 40.57, Z: 16.55 m/s^2
24   Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s
25   Temperature: 12.53 degC
26
27   Acceleration X: 1.22, Y: 40.57, Z: 16.55 m/s^2
28   Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s
29   Temperature: 12.53 degC
30 }

```

Output Serial Monitor x

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM5')

Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s  
Temperature: 12.53 degC

Acceleration X: 1.22, Y: 40.57, Z: 16.55 m/s^2  
Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s  
Temperature: 12.53 degC

Acceleration X: 1.22, Y: 40.57, Z: 16.55 m/s^2  
Rotation X: -1.30, Y: -3.27, Z: 0.00 rad/s  
Temperature: 12.53 degC

Ln 117, Col 2 DOIT ESP32 DEVKIT V1 on COM5

```

14 Serial.println("Adafruit MPU6050 test!");
15
16 // Try to in
17 if (!mpu.beg
18   Serial.pri
19   while (1)
20     delay(10
21 }
22 }
23 Serial.print
24
25 mpu.setAccel
26 Serial.print
27 switch (mpu.
28 case MPU6050
29 Serial.pri

```

Libraries

- Adafruit MPU6050
- Adafruit SSD1306
- Adafruit Unified Sensor
- ATtinySerialOut
- EspSoftwareSerial
- MPU6050
- Sim800L Library Revised
- TinyGPSPlus-ESP32

Examples from Custom Libraries

- Adafruit BusIO
- Adafruit FONA Library
- Adafruit GFX Library
- Adafruit MPU6050
- Adafruit SSD1306
- Adafruit Unified Sensor
- ATtinySerialOut
- EspSoftwareSerial
- MPU6050
- Sim800L Library Revised
- TinyGPSPlus-ESP32

basic\_readings

motion\_detection

MPU6050\_oled

mpu6050\_unifiedsensors

plotter

sleep\_demo

Ln 9, Col 19 DOIT ESP32 DEVKIT V1 on COM5

```
// Basic demo for accelerometer readings from Adafruit MPU6050

#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
    case MPU6050_RANGE_2_G:
      Serial.println("+2G");
      break;
    case MPU6050_RANGE_4_G:
      Serial.println("+4G");
      break;
    case MPU6050_RANGE_8_G:
      Serial.println("+8G");
      break;
    case MPU6050_RANGE_16_G:
      Serial.println("+16G");
      break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
    case MPU6050_RANGE_250_DEG:
      Serial.println("+ 250 deg/s");
      break;
    case MPU6050_RANGE_500_DEG:
      Serial.println("+ 500 deg/s");
      break;
    case MPU6050_RANGE_1000_DEG:
      Serial.println("+ 1000 deg/s");
      break;
    case MPU6050_RANGE_2000_DEG:
      Serial.println("+ 2000 deg/s");
      break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
    case MPU6050_BAND_260_HZ:
      Serial.println("260 Hz");
      break;
    case MPU6050_BAND_184_HZ:
      Serial.println("184 Hz");
      break;
  }
}
```

```

case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
}

Serial.println("");
delay(100);
}

void loop() {

    /* Get new sensor events with the readings */
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    /* Print out the values */
    Serial.print("Acceleration X: ");
    Serial.print(a.acceleration.x);
    Serial.print(", Y: ");
    Serial.print(a.acceleration.y);
    Serial.print(", Z: ");
    Serial.print(a.acceleration.z);
    Serial.println(" m/s^2");

    Serial.print("Rotation X: ");
    Serial.print(g.gyro.x);
    Serial.print(", Y: ");
    Serial.print(g.gyro.y);
    Serial.print(", Z: ");
    Serial.print(g.gyro.z);
    Serial.println(" rad/s");

    Serial.print("Temperature: ");
    Serial.print(temp.temperature);
    Serial.println(" degC");

    Serial.println("");
    delay(500);
}

```

پس با استفاده از سنسور و کدهای بالا ما شتاب و زاویه خودرو را تشخیص داده و در صورت تغییر ناگهانی در شتاب یا تغییر زاویه طبق دستورات کد مازول به ما پیامک اعلان خطر میدهد.

در ادامه به توضیح توابع بکارگرفته شده برای سنسور شتاب سنچ خود در کد میپردازیم.

```

245 void Impact()
246 {
247     sensors_event_t a, g, temp;
248     mpu.getEvent(&a, &g, &temp);
249     int oldx = xaxis;
250     int oldy = yaxis;
251     int oldz = zaxis;
252
253     xaxis = a.acceleration.x;
254     yaxis = a.acceleration.y;
255     zaxis = a.acceleration.z;
256
257     deltax = xaxis - oldx;
258     delty = yaxis - oldy;
259     deltz = zaxis - oldz;
260
261     magnitude = sqrt(sq(deltax) + sq(delty) + sq(deltz));
262     if (magnitude >= sensitivity) //impact detected
263     {
264         sendSms("Accident Alert!!\r\n");
265         sendLocation();
266     }
267     else
268     {
269         magnitude = 0;
270     }

```

Output

Ln 265, Col 17 DOIT ESP32 DEVKIT V1 on COM5 [not connected]

تابعی که در بالا آورده شده است تابع Impact نام دارد که وظیفه آن شتاب X,Y,Z ما را در لحظه قبل از لحظه حال کم میکند و مربعات دلتاهای بدست آمده از X,Y,Z را زیر رادیکال میبرد و اگر این مقدار از مقدار داده شده ما بیشتر بود آنگاه یک پیامک Accident Alert برای ما ارسال میکند و در ادامه لوکیشن را نیز برای ما ارسال میکند که این فرمول بدست آمده و مقدار تعیین کننده با آزمایش های مختلف انجام شده در طول پروژه بدست آمده است و این فرمول میتواند تغییر کرده و دقت آن بالاتر برود.

همچنین کد زیر، پیامک را در صورت برخورد به شماره ای که از قبل ثبت کردیم ارسال میکند.

```

208 delay(1000);
209
210 SIM800.println("AT+CMGS=\"+989219656927\""); //Send SMS to this number
211 delay(1000);
212
213 SIM800.println(Link);
214 delay(1000);
215
216 SIM800.println((char)26);
217 delay(1000);
218
219 }
220 void sendSms(String text)
221 {
222     sim800.print("AT+CMGF=1\r\n");
223     delay(1000);
224     sim800.print("AT+CMGS=\"\" + PHONE + \"\"\r\n");
225     delay(1000);
226     sim800.print(text);
227     delay(100);
228     sim800.write(0x1A);
229     delay(1000);
230     Serial.println("SMS Sent Successfully.");
231 }
232

```

Output

Sketch uses 275781 bytes (21%) of program storage space. Maximum is 1310720 bytes.

Ln 215, Col 1 DOIT ESP32 DEVKIT V1 on COM5 [not connected]





## Sending Data to Sever

در این بخش پس از توضیح نحوه اتصالات و کدها بطور مختصر به سراغ نحوه ارسال داده به سرور برای مانیتور کردن و کنترل خودرو میپردازیم.

پروتکل های مطرح ارسال و دریافت دیتا

- http انتقال داده به سرور
- ftp انتقال فایل به سرور
- Smtip ارسال و دریافت ایمیل

ماژول SIM800 و SIM800L این ۳ نوع پروتکل رو پشتیبانی میکند.

یکی از قابلیت های جالب و جذاب ارسال دیتا به سرور از طریق اینترنت توسط SIM800 می باشد. برای اتصال ماژول به GPRS باید حتما سیم کارت دارای شارژ باشد. ماژول SIM800 قابلیت این را دارد که بصورت کلاینت و سرور عمل نماید. برای ارتباط بین کلاینت و سرور باید یک اتصال TCP/IP برقرار شود. به این ارتباط سوکت نیز میگویند. TCP/IP در واقع مثل یک پل ارتباطی بین کلاینت و سرور هست که باعث میشه از طرف ما که کلاینت باشیم به سرور دیتا ارسال کنیم. برای ایجاد اتصال TCP/IP ما نیاز به آدرس سروس یا IP اون و شماره پورت داریم. شماره پورت ۲۵ برای پروتکل SMTP، شماره پورت ۲۱ برای FTP و شماره پورت ۸۰ برای http رزرو شده است. برای برقراری ارتباط بین کلاینت و سرور نیاز به ایجاد ارتباط TCP/IP است. در HTTP روش های ارسال درخواست به سرور بنام های GET، SEND، HEAD، POST، LINK و ... وجود دارد. از بین چندین موردی که ذکر شد، ماژول های SIM800 از سه متد HEAD، GET و POST پشتیبانی می کند.

برای ارسال و دریافت داده در سطح اینترنت، پروتکل های متنوع بسته به کارکرد وجود دارند. ماژول SIM800L قادر است تا از اکثر پروتکل های اینترنتی، پشتیبانی نماید. به عنوان مثال، به کمک پروتکل FTP، می توانید فایل آپلود و یا دانلود نمایید. به کمک پروتکل SMTP به ارسال و دریافت ایمیل بپردازید. همچنین، به کمک پروتکل HTTP می توانید داده ها را به روش POST و یا GET نمایید. به کمک قابلیت GPRS ماژول SIM800L، می توان پروژه های اینترنت اشیا را بدون وابستگی به مودم وای فای و یا مودم دیگر، اجرا کرد. با توجه به قابلیت های ماژول SIM800L که قادر به ارسال داده های محیطی نظیر دما، رطوبت، فشار و ... است، پروژه های اینترنت اشیا را می توان با سرعت و هزینه کم اجرا نمود. حتی با قابلیت FTP این ماژول، می توان به ارسال فایل و دانلود فایل نیز پرداخت.

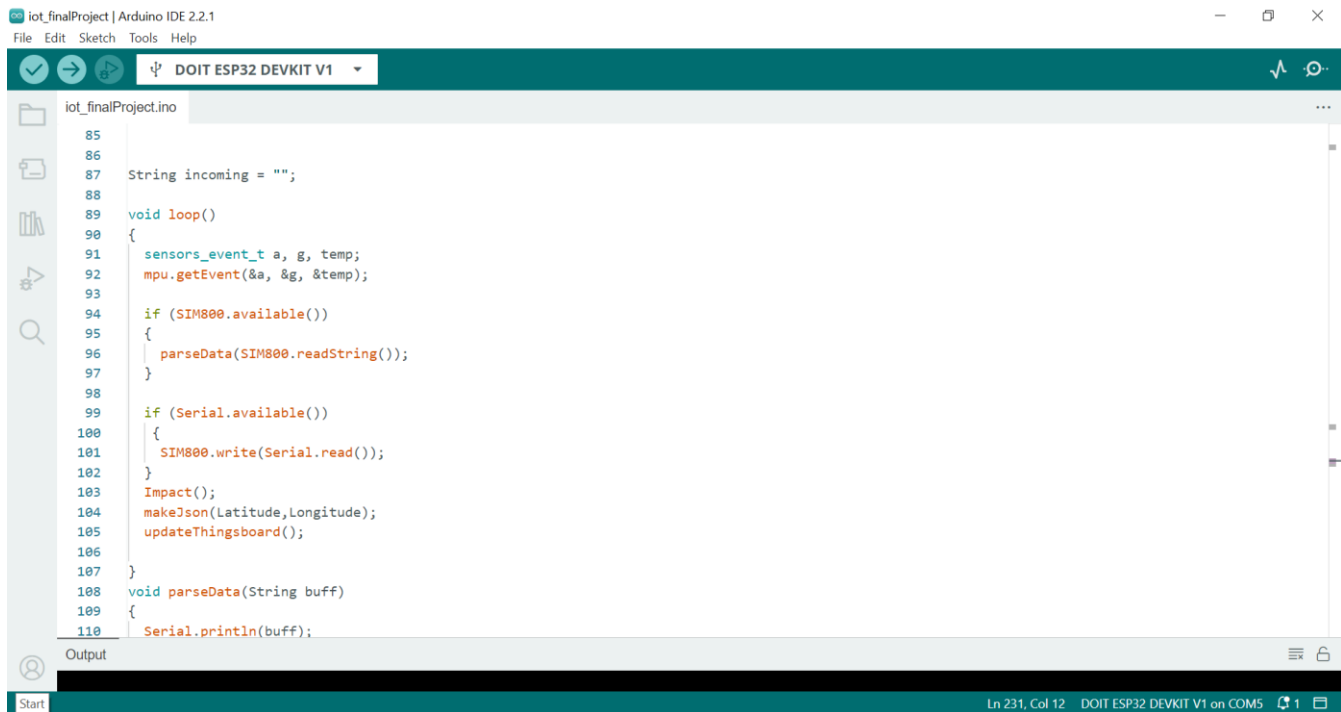
در ادامه به سراغ کد های بخش ارسال داده به پلتفرم میرویم. نکته ای که حائز اهمیت است این است که در تمامی این کد ها ما باید از کد های AT Commands استفاده کنیم که برای جلوگیری از طولانی شدن این گزارش از آوردن آنها خودداری شده است. این کد ها دستورات لازم را به ماژول برای ارسال داده به پلتفرم میدهند.

```
String AccessToken = "4pilpavu80qobbafh8i3";

char thingsboard_url[] = "mqtt.thingsboard.cloud";

makeJson(float(Latitude), float(Longitude));
updateThingsboard();
```

برای آپدیت کردن موقعیت مکانی ما در پلتفرم بصورت لحظه به لحظه از تابع `updateThingsboard` استفاده کرده و همچنین `Access Token` در کد بصورت های بال تعریف شده اند. همچنین خط سوم و چهارم را در `loop` اصلی قرار دایم تا در هر لحظه لوکیشن را داشته باشیم.



```
iot_finalProject.ino
85
86
87 String incoming = "";
88
89 void loop()
90 {
91   sensors_event_t a, g, temp;
92   mpu.getEvent(&a, &g, &temp);
93
94   if (SIM800.available())
95   {
96     parseData(SIM800.readString());
97   }
98
99   if (Serial.available())
100  {
101    SIM800.write(Serial.read());
102  }
103  Impact();
104  makeJson(Latitude, Longitude);
105  updateThingsboard();
106
107 }
108 void parseData(String buff)
109 {
110   Serial.println(buff);
111 }
```

در این بخش از کد نیز همانگونه که مشاهده میشود تابع `updateThingsboard` را داخل `loop` اصلی برنامه قرار می دهیم تا در هر لحظه لوکیشن را داشته باشیم.





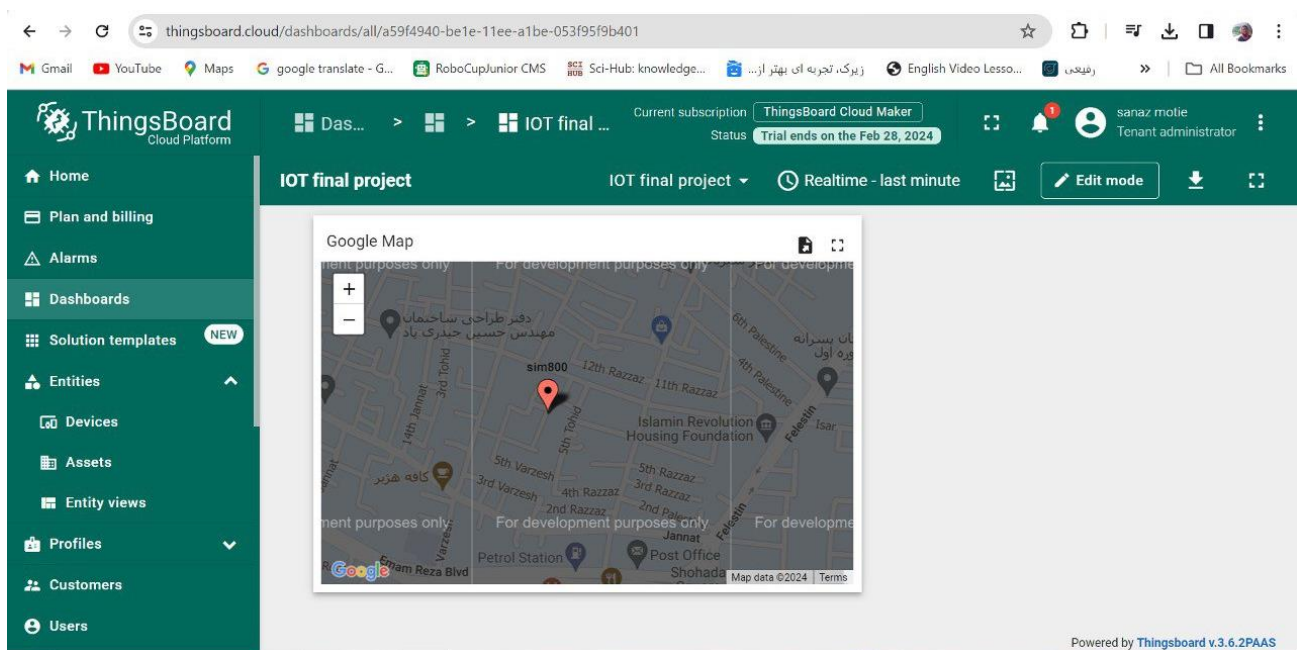
## IoT Platform

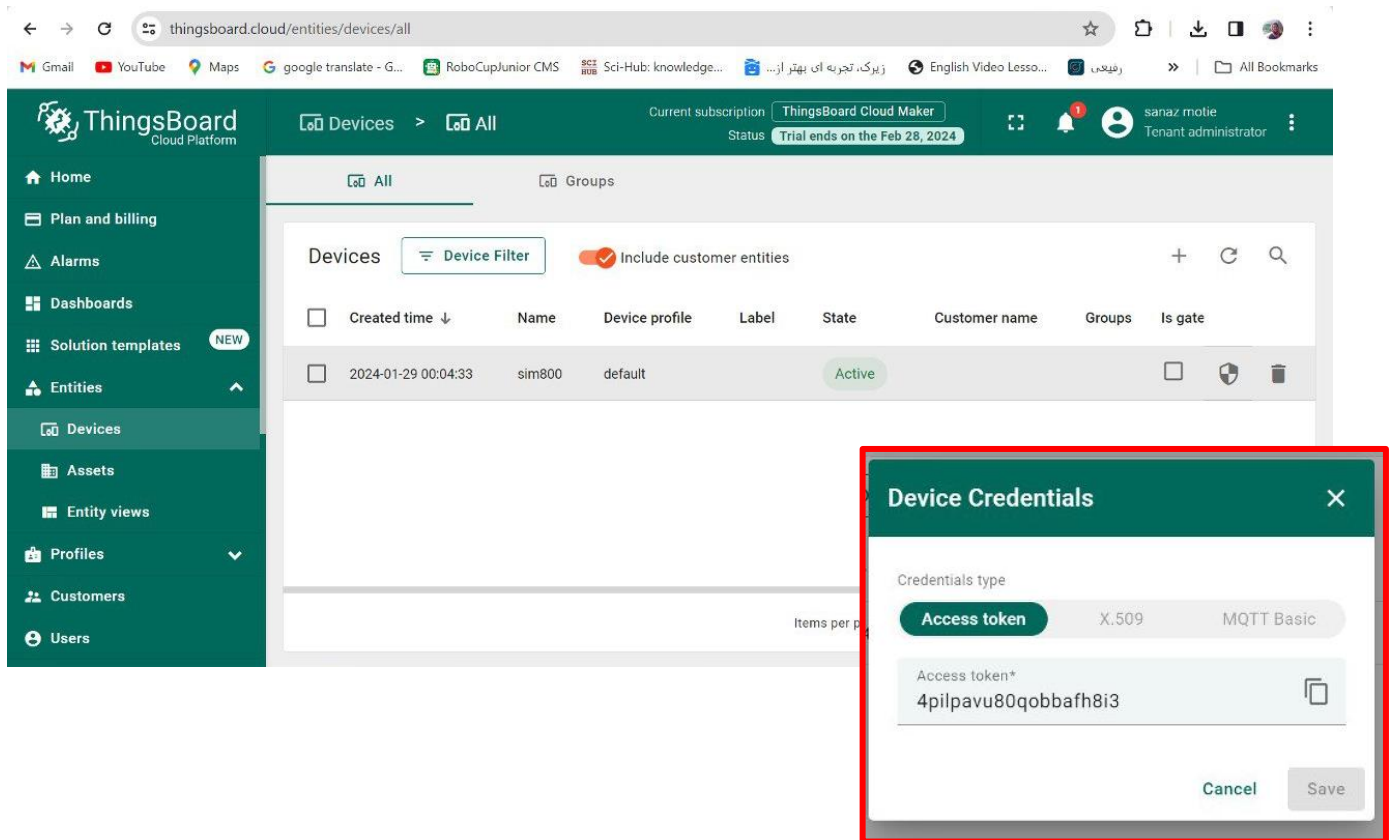
ما علاوه بر کنترل راننده و خودرو از طریق پیامک نیاز به مانیتور کردن آن در لحظه و مشاهده موقعیت مکانی و سرعت آن در هر لحظه داریم. بنابراین ما نیاز به یک پلتفرم داریم که ماژول بتواند دیتا را برای آن ارسال کند و ما بتوانیم دیتا را دریافت و مانیتور کنیم. پلتفرمی که ما برای این پروژه در نظر گرفتیم ThingsBoard است.



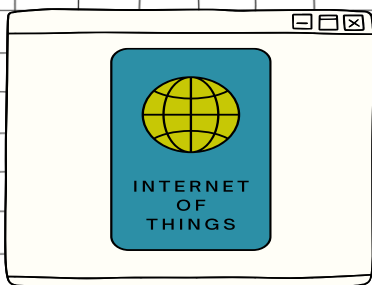
همانطور که در بخش قبل گفته شد ماژول ما توانایی ارسال داده را داراست و میتواند داده را به سرور بفرستد، پس راه حل ما برای مانیتور کردن دیتای ارسالی راننده به این شیوه بود که داده ها از طریق ماژول به سرور ارسال میشد و سپس از طریق پلتفرم ما مانیتور و کنترل میشد. در ادامه به شیوه دستیابی به دیتاهای ارسالی از ماژول و مانیتور کردن آن بر روی پلتفرم میپردازیم.

همانطور که مستحضر هستید در کلاس درس در مورد این پلتفرم مفصل توضیح داده شده است پس در این گزارش از اضافه گویی خودداری می‌شود. مطابق شکل های زیر در پلتفرم ما و در قسمت Entities و در بخش Devices یک Device جدید به اسم sim800 ساخته و در Dashboards یک Google Maps قرار میدهم که موقعیت جغرافیایی ما را از دیتای دریافتی ماژول میخواند و آن را نمایش میدهد. در عکس دیگر Access Token را مشاهده میکنیم که برای هر کاربر منحصر به فرد است برای اینکه بتوانیم یک ارتباط امن داشته باشیم و امنیت داده های خود را تضمین کنیم.





پس همانگونه که در تصاویر مشاهده شد ما از طریق ارسال دیتا از مژول به پلتفرم خود توانیستیم Real-Time لوکیشن خود را مانیتور کنیم و تغییرات آن را مشاهده کنیم.





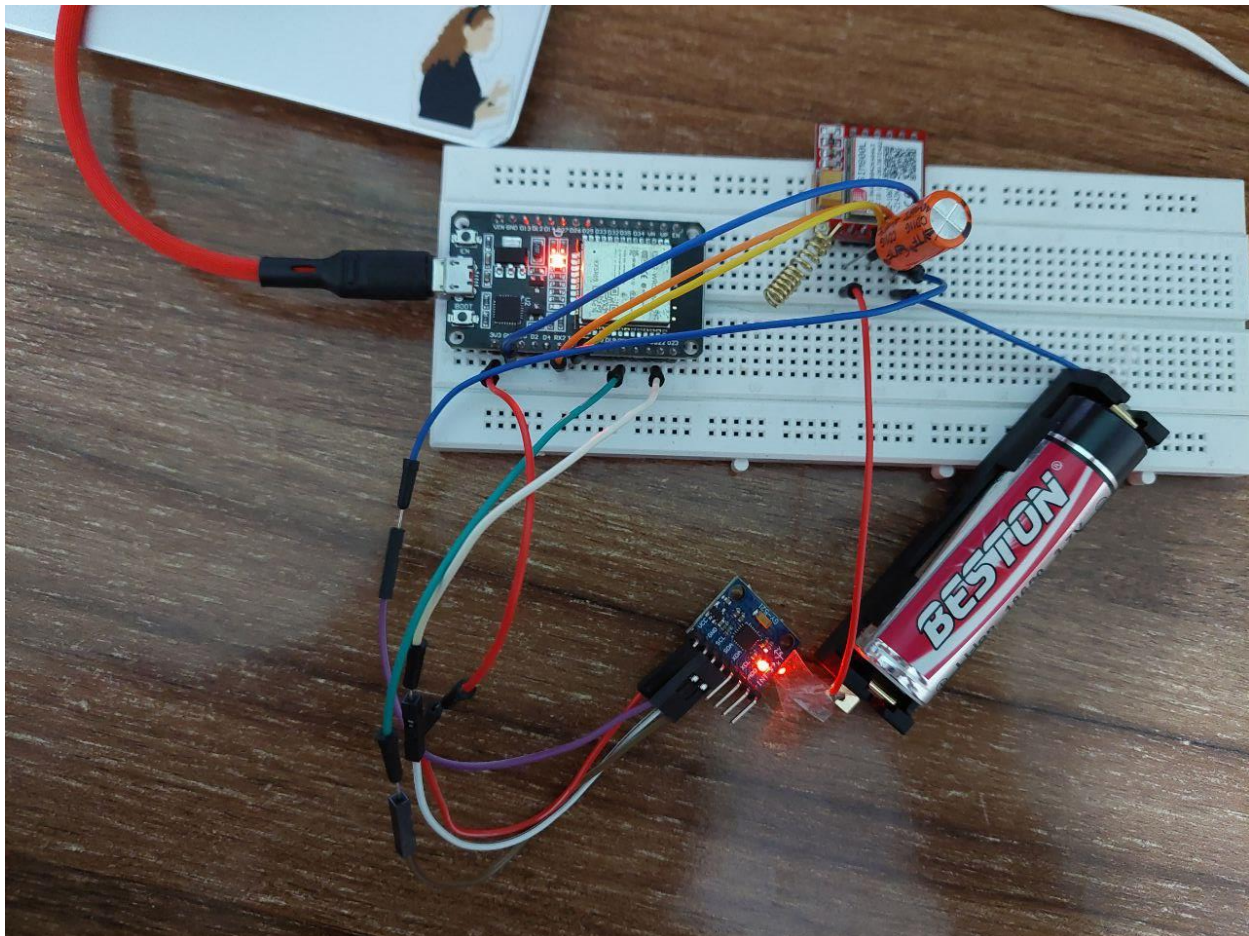
## Conclusion

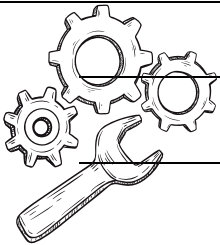
با توجه به آنچه تا کنون از روند انجام پروژه و مشخصات و ویژگی های ماژول ها و سنسورها و همچنین توضیحات مختصری از کد گفته شد ما توانستیم با حدود ۶۰۰ هزار تومان تمام وسایل لازمه را خریداری و یک راه حل IoT برای مشکل مانیتورینگ و کنترل رانندگان ناوگان لاجستیک ارائه دهیم.

در این پروژه ما از ماژول Sim800l استفاده کردیم که توانستیم لوکیشن رانندگان را بطور لحظه ای مورد بررسی قرار دهیم و همچنین در هر لحظه از طریق پیامک لوکیشن آنها را بدست آوریم و از خطرات حین رانندگی از جمله تصادفات آگاه بشویم. همچنین از طریق پلتفرم ThingsBoard این مانیتورینگ را انجام دادیم که البته گزینه های دیگری نیز برای پلتفرم ما نظیر Node-red, Thingspeak وجود داشت.

در نهایت در این پروژه ما به یک ویژگی اساسی این پروژه های IoT یعنی پرتابل بودن دست پیدا کردیم که از نظر من یک ویژگی اساسی در این نوع پروژه ها است. ما توانستیم با استفاده از یک باتری و چند سنسور و ماژول GPS Tracker خود را ساخته و در آینده ویژگی های کاربردی بیشتری نیز به آن اضافه کنیم.

در ادامه مدار نهایی پروژه ما آورده شده است:





## Future Features

در نهایت این پروژه را میتون بسط داد و ویژگی های بیشتری به آن اضافه نمود که در این بخش توضیح مختصری از آنچه در ذهن ما بود و متاسفانه به دلیل کمبود وقت و امکانات عملی نشد داده میشود.

این پروژه قابلیت های بسیاری برای بسط دادن دارد که میتوان از جمله آنها افزودن سنسور دما برای مانیتور دمای درون کابین برای جلوگیری از فساد مواد استفاده کرد بطوری که ما همزمان دما را مانیتور کرده و اگر از محدوده تعیین شده کمتر یا بیشتر شد با ارسال پیامک به ما هشدار بدهد. همچنین همین مورد را میتوان با افزودن یک سنسور رطوبت برای بررسی میزان رطوبت مانند دما بسط داد که همه ی این اخطارها همراه با یک بوق برای هشدار دادن همزمان به راننده نیز باشد. از ویژگی هایی که میتوان به این پروژه افزود افزودن حد سرعت برای کنترل سرعت راننده و مانیتور کردن آن بر روی پلتفرم مورد نظر ما و همچنین تماس با اپراتور در صورت عبور از حد مجاز سرعت و همچنین ارسال پیامک برای دریافت سرعت لحظه ای خودرو میباشد، ویژگی جذاب دیگر کنترل راننده برای عدم خروج از محدوده تعیین شده توسط اپراتور است که در صورت تخطی یک تماس گرفته خواهد شد و در صورت عدم پاسخگویی به تماس یک پیام خطر با مضمون عبور راننده از مرز تعیین شده در موقعیت جغرافیایی میباشد.

در نهایت میتوان این پروژه را با هوش مصنوعی تلفیق نموده و با جایگذاری یک دوربین خواب آلودگی راننده را تشخیص داده و به اپراتور با ارسال پیامک اطلاع دهد یا در روشی دیگر با یادگیری الگوریتم سرعت راننده و یا حرکت آن تشخیص دهد که راننده خواب آلود است یا خیر و این پروژه را به یک پروژه عملی قابل فروش تبدیل کرد.





## References

با تشکر از توجه و مطالعه شما، در ادامه منابع مورد استفاده بنده در ساخت این پروژه را آورده ام.

- <https://micronik.ir/sim800-mqtt/>
- <https://digispark.ir/sim-card-module-gprs-internet-sim800l/>
- <https://micronik.ir/sim800-gprs/>
- <https://thingsboard.io/>
- <https://www.youtube.com>