

Smart Anti-Theft Solution for Vehicles

A Project submitted to the faculty of
San Francisco State University
In partial fulfillment of
the requirements for
the Degree

Master of Science

In

Electrical and Computer Engineering

by

Anuj Sachin More

San Francisco, California

December 2024

Copyright by
Anuj Sachin More
2024

Certification of Approval

I certify that I have read Smart Anti-Theft Solution for Vehicles by Anuj Sachin More, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirement for the degree Master of Science Electrical and Computer Engineering at San Francisco State University.

Xiaorong Zhang, Ph.D.
Associate Professor,
Project Committee Chair

Zhuwei Qin, Ph.D.
Assistant Professor

Abstract

This project presents an innovative Vehicle Tracking and Theft Prevention System designed to enhance vehicle security and provide real-time location monitoring capabilities for vehicles. The goal of this project is to develop a theft prevention system for vehicles by overcoming limitations of existing systems by incorporating remote immobilization using a simple light-weight SMS interface. The system integrates the STM32 NUCLEO-L476RG microcontroller, GPS NEO-7M, and SIM7600A-H modules for precise tracking, remote immobilization, and alert notifications via SMS. It addresses key challenges such as limited GPS antenna capabilities and network compatibility, ensuring reliable operation in diverse environments, making it ideal for personal vehicle security. Through modular testing and efficient buffer management, the system delivers efficient performance and scalability. Given the scalability this project has potential applications in the car rental and leasing business for monitoring and fleet management. Potential future enhancements include advanced diagnostic capabilities, AI-driven theft prediction algorithms, and seamless integration with smart home ecosystems.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Zhang for their invaluable guidance and support throughout the course of this project. Their expertise and constructive feedback have been instrumental in shaping the outcomes of this work. I am deeply thankful to my family for their unwavering support and understanding, which provided me with the strength and motivation to persevere. Lastly, I would like to thank my friends for their encouragement and assistance, this project would not have been possible without the contributions and support of all these wonderful individuals.

Table of Contents

Smart Anti-Theft Solution for Vehicles	i
Certification of Approval	vi
Abstract.....	vii
Acknowledgements	v
Table of Contents	Error! Bookmark not defined.
List of Tables	vii
List of Figures.....	vii
Introduction.....	1
Related Works	3
Current Limitations.....	4
Design and Implementation of Smart Anti-Theft Solution for Vehicles	5
System Overview	5
Hardware Components.....	7
Software Implementation.....	13
SMS based User Interface.....	21
Results	22
GPS Module Performance Results.....	22
4G Module Performance Result.....	25
Demonstration of Overall Working	26
Discussion.....	28
Future Work	28
Applications	29
Conclusion	30
References	31
Appendix/Appendices	33

List of Tables

Table 1. Statewise Number of stolen vehicles in 2022 and 2023	1
Table 2. Specifications of STM32 NUCLEO-L476RG	8
Table 3. Specifications of GPS NEO-7M	9
Table 4. Specifications of SIM7600A-H Module.....	10
Table 5. Specifications of the 5V Relay Module.....	11
Table 6. Pin-Pin Hardware Connections	12

List of Figures

Figure 1. US motor vehicle theft rate from 2018 to 2023 by Axios	2
Figure 2. Block diagram of the smart anti-theft solution for vehicles	5
Figure 3. Physical & Pinout diagram of STM32-NUCLEO-L476RG	7
Figure 4. Physical and Pinout diagram of GPS NEO-7M Module.....	8
Figure 5. Physical diagram of SIM7600A-H Module	10
Figure 6. Physical and Pinout Diagram of 5V Relay module	11
Figure 7. Hardware Interface Design with connections	11
Figure 8. GPRMC Sentence Format	15
Figure 9. Raw Data from GPS Module.....	16
Figure 10. Extracted Data using extractGPRMCData().	17
Figure 11. AT commands to Receive Message	18
Figure 12. AT commands to Send Message Commands	19
Figure 13. Screenshots of SMS interface.	22
Figure 14. Time taken to get GPS Fix	24
Figure 15. Time taken to read a new message and respond to it.....	25
Figure 16. Image taken when car is static and Lock command is sent to system	26
Figure 17. Image taken when car is moving	27

Introduction

Vehicle theft is a growing problem which requires better and economic, security solutions to protect both personal and commercial vehicles. Motor vehicle theft continues to be significant issue across the United States, with California leading as the state with the highest number of vehicle thefts in 2023. According to an article by the Insurance information Institute, in 2023 alone, over 208,000 vehicles were stolen in California, marking a 3% increase from 2022 [1].

Table 1. Statewise Number of stolen vehicles in 2022 and 2023

Rank	State	Year 2023	Year 2022
1	California	208,668	202,685
2	Texas	115,013	105,015
3	Florida	46,213	45,973
4	Washington	43,160	46,939
5	Illinois	41,528	38,649
6	Colorado	34,068	42,237
7	New York	32,715	28,292
8	Ohio	31,647	29,913
9	Georgia	28,171	26,529
10	Missouri	27,279	29,345

Another article from axios.com [2] indicates The United States as a whole has experienced a steady rise in motor vehicle theft rates, with thefts per 100,000 residents climbing sharply between 2018 and mid-2023.

U.S. motor vehicle theft rate

Thefts per 100k residents; Monthly, January 2018 to June 2023



Figure 1. US motor vehicle theft rate from 2018 to 2023 by Axios

California, despite its advancements in vehicle security, also ranks among the top states in theft rate, with a rate exceeding 520 thefts per 100,000 residents in 2022. This statistics highlight the need for reliable and innovative anti-theft solutions, especially in hotspots like California. This project addresses this challenge by proposing a Vehicle Anti-Theft System that uses modern technologies like ARM microcontrollers, GPS, GSM communication, and advanced theft detection to offer enhanced security. The system is designed to track your vehicle's location in real time, alert you immediately if someone tries to steal it, and immobilize the vehicle to prevent further theft. This project leverages the cost-effectiveness and availability of these technologies to make advanced vehicle protection accessible to a wide audience. It includes a simple SMS based interface that allows you to monitor and control the system remotely. By focusing on reliability, ease of use, and thorough testing, this proposal aims to provide a practical and effective solution to reduce vehicle theft and enhance overall vehicle security.

Related Works

In recent years, various IoT-based vehicle security systems have been developed to provide protection against theft and ensure quick response in the event of accidents. An example is the IoT-based Anti-Theft System proposed by Balakrishnan and Murugan uses the Node-MCU, an ultrasonic sensor, and an ESP32 camera for detecting theft, capturing images, and notifying the owner via e-mail [3]. Similarly, the use of GSM and GPS modules for real-time tracking and remote vehicle control is explored in the anti-theft system for vehicles by B. Tech students at D.M.S.S.V.H. College [4], which features OTP-based security and live location tracking. Another ARM Based Vehicle Theft Control, Positioning, and Collision Detection System [5], incorporates advanced security features such as fingerprint verification and GPS-based tracking, aimed at preventing theft and providing real-time updates in case of accidents. These studies emphasize the importance of integrating multiple sensors, communication modules, and real-time monitoring to create robust security systems for vehicles, which can effectively prevent theft and aid during emergencies.

Some current vehicle anti-theft technologies incorporate a variety of physical, electronic, and biometric solutions aimed at enhancing vehicle security. Mechanical steering wheel locks which are most widely used, such as the Zclub 300, provide physical protection to theft, are priced between \$30 and \$60, making them an affordable yet effective choice. Car alarms, offered by brands like Viper and Clifford, are used to trigger audible and visual alerts in the event of a breach. Biometric systems, like the fingerprint recognition feature in vehicles by Hyundai and Kia, ensure only authorized users can access the vehicle, adding an extra layer of security.

Immobilizers, such as the Ravelco Anti-Theft Device, prevent unauthorized engine access by requiring a matching code, with prices ranging from \$200 to \$500. GPS tracking systems, including solutions like LoJack and Vyncs, provide real-time vehicle tracking and diagnostics, significantly enhancing the chances of recovery in case of theft, with prices varying from \$100 to \$1000. These technologies offer varied levels of protection, combining traditional mechanical methods with advanced electronic systems to address the growing concerns of vehicle theft.

Current Limitations

Current vehicle anti-theft solutions are innovative but often present significant challenges regarding installation, usability, and overall reliability. Many systems require complex installations, typically involving professional setup or meticulous configurations, which increases both the cost and time commitment for vehicle owners along with the need for an expert for the task.

This complexity can discourage vehicle owners from adopting these solutions, particularly among those seeking simple, user-friendly solutions. Furthermore, a number of systems rely solely on mobile apps for vehicle monitoring and control, which can be problematic during server downtimes or instances where connectivity is lost. In such situations, the user loses access to critical features, leaving the vehicle exposed to theft.

Another key limitation is the lack of an SMS-based interface, which does not rely on internet connectivity or app functionality, making it a much more resilient option in emergencies, especially in areas with weak or no network coverage. Furthermore, while many systems include

tracking features, they generally lack remote immobilization capabilities, preventing users from disabling their vehicle remotely if it is stolen. This means that even when a theft is detected, the owner has limited options for preventing the vehicle from being driven away. These limitations highlight the need for more flexible, resilient, and comprehensive anti-theft solutions that can provide better security, ease of use, and control for vehicle owners.

Design and Implementation of Smart Anti-Theft Solution for Vehicles

System Overview

Block Diagram

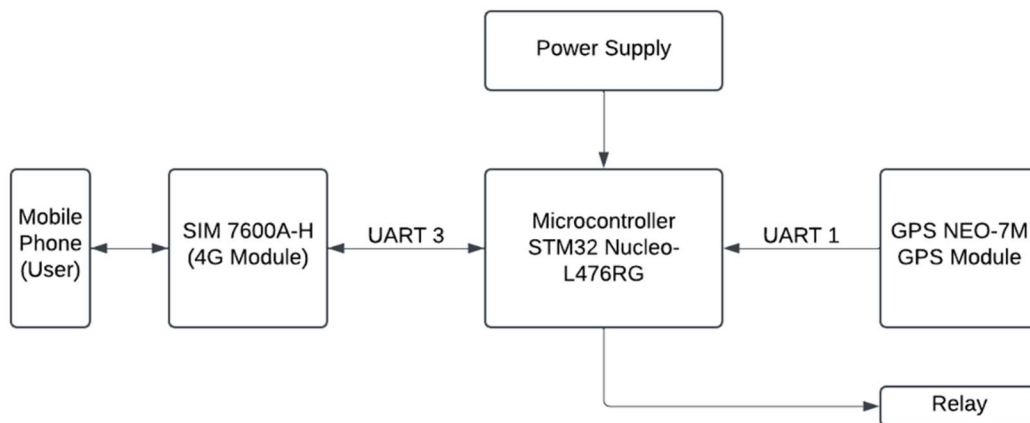


Figure 2. Block diagram of the smart anti-theft solution for vehicles

The block diagram illustrates the overall design of the vehicle tracking and theft prevention system. The STM32 Nucleo-L476RG microcontroller acts as the central control unit, interfacing with the GPS NEO-7M module via UART1 to receive real-time location data. A SIM7600A-H that is the 4G module is connected to the microcontroller through UART3. This

module enables communication with the user's mobile phone for alerts and commands via SMS.

The system uses a dedicated power supply to ensure stable operation. For actuation purposes, the microcontroller controls a relay circuit for immobilizing the vehicle by breaking the ignition circuit when theft is detected or upon owner command. The interconnections between the components provide a seamless integration of GPS tracking, GSM communication, and vehicle control functionalities, ensuring a reliable theft prevention mechanism..

System Workflow

The system workflow begins with the initialization phase, where the microcontroller, GPS module, and GSM module are set up and configured to ensure proper communication and functionality. Once initialized, the GPS module provides continuous real-time location data to the microcontroller for tracking purposes. In the theft detection phase, the microcontroller monitors the location data to identify any unauthorized movement or deviations. If theft or suspicious activity is detected, the GSM module sends an SMS alert to the owner, notifying them of the location change. To further secure the vehicle, the system can immobilize it by activating a relay that breaks the ignition circuit. Finally, in the monitoring and recovery phase, the owner can track the vehicle's real-time location and take appropriate actions to recover it, leveraging the system's communication and control capabilities. This streamlined workflow ensures a robust and efficient theft prevention and vehicle tracking solution.

Hardware Components

STM32 – NUCLEO – L476RG Micro-Controller

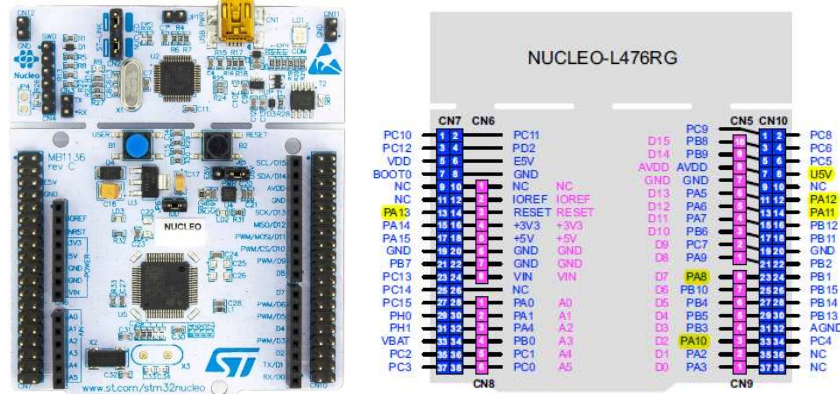


Figure 3. Physical & Pinout diagram of STM32-NUCLEO-L476RG

The STM32 NUCLEO-L476RG microcontroller is the core of the vehicle tracking and theft prevention system that manages all critical operations. It is equipped with 5 UARTs, enabling simultaneous communication with multiple peripherals, such as the GPS NEO-7M and SIM7600A-H GSM modules. The availability of 2 DMA (Direct Memory Access) controllers further enhances its efficiency by allowing high-speed data transfers between peripherals and memory without burdening the CPU, making it ideal for real-time applications.

Additionally, the microcontroller features multiple timers which can be utilized for precise timing analysis and periodic tasks, such as monitoring GPS data or controlling system intervals. Its GPIO pins add further versatility, enabling direct control of external actuators such as the relay used for vehicle immobilization. These GPIOs provide a fast and reliable interface for triggering actions based on theft detection logic.

The STM32 NUCLEO-L476RG has ultra-low power consumption, which ensures sustained operation in battery-powered systems. Its compatibility with various communication

protocols, including UART, SPI, and I2C, allows for seamless integration with additional modules or sensors. The STM32's ecosystem also features STM32 HAL libraries which has good developer support and simplifies development, making it an excellent choice for this project. Some important specifications of the STM32 NUCLEO-L476RG which were considered for this application are mentioned in Table 2.

Table 2. Specifications of STM32 NUCLEO-L476RG

Specification	Details
Core	ARM Cortex-M4 CPU with FPU, 80 MHz
Operating Voltage	1.71 V to 3.6 V
Power Modes	VBAT, Shutdown, Stand-By, Run mode
Peripherals	5 configurable UART's, I2C, SPI
DMA	2 configurable DMA controllers

GPS NEO – 7M Module

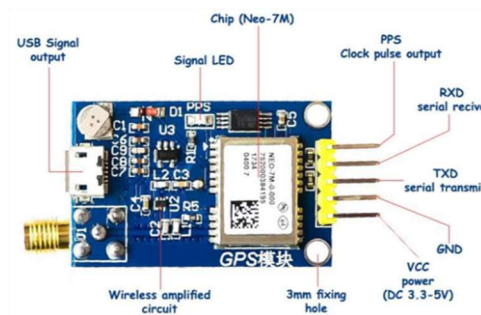


Figure 4. Physical and Pinout diagram of GPS NEO-7M Module

The GPS NEO-7M module is a compact and high-performance positioning device used for real-time vehicle tracking in the system. It offers a positional accuracy of up to 2.5 meters and operates with a default 1 Hz update rate, configurable for faster updates. With its built-in UART interface, it ensures seamless communication with the STM32 NUCLEO-L476RG

microcontroller. The module is designed for low power consumption and a wide temperature range (-40 °C to +85 °C), making it suitable for automotive applications. Considered Specifications are mentioned below in Table 3.

Table 3. Specifications of GPS NEO-7M

Specification	Details
Positional Accuracy	2.5 meters
Navigation Update rate	1Hz (10Hz Max)
Capture Time	Cool start: 27sHot start: 1s
Serial Baud rate	4800-230400 (default 9600)
Operating Temp.	-40°C to 85°C
Operating Voltage	2.7V to 3.6V
Operating Current	35mA

SIM 7600A-H 4G Module

The SIM7600A-H 4G module is a communication module used in the system to enable reliable data transmission and remote control via SMS. It supports multiple communication protocols, including 4G LTE, ensuring fast and stable connectivity. The module interfaces seamlessly with the STM32 NUCLEO-L476RG microcontroller through its UART interface, enabling real-time notification and command functionalities. With its low power consumption and wide operating temperature range (-30 °C to +85 °C), it is well-suited for automotive and IoT applications. The SIM7600A-H also features built-in GNSS support, enhancing its functionality for location-based services, making it an integral component for secure and efficient vehicle monitoring and theft prevention.



Figure 5. Physical diagram of SIM7600A-H Module

Certain specifications of this 4G Module which make it a good fit for this particular application are mentioned below in Table 4.

Table 4. Specifications of SIM7600A-H Module

Specifications	Details
Operating Frequency	B2/B4/B12 (Freq. Bands for USA)
Operating Temp.	-30°C to +85°C
Operating Voltage	3.3V to 5V
Control Method	AT Commands using UART
Serial Baud rate	115200 (default)
External Antenna compatibility	Enabled

Relay Module

The 5V relay module is the actuation component in the system that is responsible for controlling the vehicle's ignition circuit to achieve immobilization. It operates as an electrically controlled switch which is triggered by the STM32 NUCLEO-L476RG microcontroller through its GPIO pins. The relay provides electrical isolation and can handle high-current loads, ensuring safe and reliable operation. Its 5V operating voltage ensures compatibility with the

microcontroller. Additionally, the relay module has been selected with future scalability in mind, as it can potentially be integrated with the vehicle's ECU (Engine Control Unit) to enable more advanced control and security features, enhancing the system's overall functionality.



Figure 6. Physical and Pinout Diagram of 5V Relay module

Some important specifications of this relay module which were considered for this application are given in Table 5 below.

Table 5. Specifications of the 5V Relay Module.

Specification	Details
Relay Control Voltage	5 Volts
Trigger Current	15mA to 20mA
DC Load Capacity	Up to 30V at 10 A

Hardware Interface Design

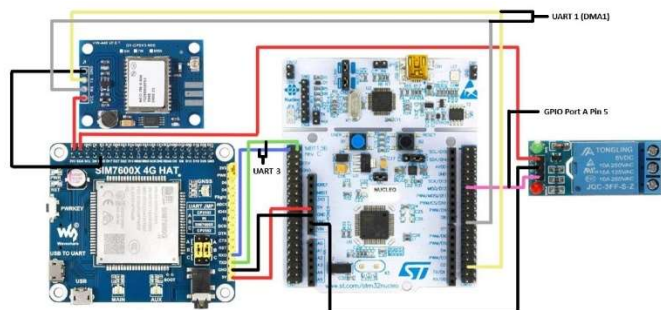


Figure 7. Hardware Interface Design with connections

The hardware interface design, as shown in Figure 7, integrates multiple modules with the STM32 NUCLEO-L476RG microcontroller. The GPS NEO-7M module is connected to the microcontroller using UART1 in DMA mode, utilizing the DMA1 controller for data transfer without CPU intervention. The RX and TX pins of UART1 are mapped to Port A pins PA10 and PA9, respectively, for real-time location data acquisition. Similarly, the SIM7600A-H 4G module is connected to the microcontroller via UART3, with the RX and TX pins assigned to Port C pins PC10 and PC11, enabling communication for sending SMS alerts and receiving commands. The relay module is interfaced with the microcontroller through its signal pin, which is connected to GPIO Port A Pin 5 (PA5). This GPIO pin is used to trigger the relay for vehicle immobilization when theft is detected. These hardware connections ensure signal flow, efficient communication, and accurate control across all modules in the system without any intervention of each other with the microcontroller.

Table 6. Pin-Pin Hardware Connections

Module	Module Pin	STM32 Pin	Purpose
GPS Neo-7M	RX	Port A Pin 9	UART 1(TX)
GPS Neo-7M	TX	Port A Pin 10	UART 1(RX)
SIM7600A-H	RX	Port C Pin 10	UART 3(TX)
SIM7600A-H	TX	Port C Pin 11	UART 3(RX)
Relay	Signal	Port A Pin 5	Control O/P

Software Implementation

The software implementation of this project focuses on integrating hardware components and ensuring efficient system operation based on the desired functionality. The STM32CubeIDE, an integrated development environment tailored for STM32 microcontrollers, for firmware development and debugging is used. STM32 HAL (Hardware Abstraction Layer) libraries are employed to simplify low-level hardware interfacing and provide a robust software foundation. Key software components include microcontroller interfacing, where the STM32 NUCLEO-L476RG is programmed to manage all peripherals.

GPS communication is implemented using UART1 peripheral for continuous and efficient reception of real-time location data from the NEO-7M GPS module. Similarly, GSM communication is handled through UART3, enabling the SIM7600A-H module to send SMS alerts and receive commands. These components work together cohesively to detect theft, trigger vehicle immobilization, and provide reliable communication, ensuring the system operates smoothly and meets its functional objectives.

Microcontroller Interfacing

The Software Implementation to interface the other modules with the microcontroller involved the initialization, configuration, and management of various peripherals on the STM32 NUCLEO-L476RG microcontroller to ensure proper communication, data handling, and actuation within the system. The `SystemClock_Config()` function is used to configure the system clock using the High-Speed Internal (HSI) oscillator (16MHz default frequency). This HSI oscillator frequency is given to PLL which using the PLLM, PLLN and PLLR to output 80Mhz as the PLLCLK which is used as System Clock Source. For UART Communication, the

MX_USART1_UART_Init() and MX_USART3_UART_Init() functions were used to initialize USART1 and USART3 for communication with the GPS NEO-7M and SIM7600A-H 4G modules, respectively. USART1 was configured with a baud rate of 9600 for GPS data reception which is the default baud rate for GPS module. Further, HAL_UART_Receive_DMA() function is used to enable non-blocking communication via DMA1 channel. Similarly, USART3, with a baud rate of 115200 that is the default baud rate for 4G Module , handled 4g module communication using HAL_UART_Transmit() and HAL_UART_Receive() for sending and receiving SMS data. The HAL_UART_RxCpltCallback() interrupt function was implemented to process data when UART reception was complete, ensuring reliable real-time handling of GPS responses.

The GPIO Configuration was managed in MX_GPIO_Init(), where Port A Pin 5 (PA5) was initialized as an output pin in push-pull mode to control the relay module. The relay was initially set to LOW using HAL_GPIO_WritePin(), ensuring it remained inactive until explicitly triggered for vehicle immobilization. For timing analysis and periodic tasks, MX_TIM16_Init() was configured to initialize Timer 16 with a Prescaler = 0 in Up-count mode, which is utilized for precise timing requirements. Additionally, MX_DMA_Init() enabled DMA controller clock and configured interrupt priorities for UART1 communication.

These configurations, combined with the use of STM32 HAL library functions such as HAL_UART_Transmit/Receive(), HAL_UART_Transmit/Receive_DMA(), and HAL_UART_RxCpltCallback(), ensured efficient communication, real-time data acquisition, and smooth actuation. Together, these implementations facilitated the core interfacing with the

GPS, 4G, and relay modules, achieving the project's functional objectives. Please refer to Appendix to see the initialization and configuration parts of the code

GPS Communication

In the GPS Communication section of the software implementation, continuous data reception from the NEO-7M GPS module was achieved using the `HAL_UART_Receive_DMA()` function. This function was called within the main loop to initialize DMA-based UART communication on USART1, configured to receive GPS data in a non-blocking manner. Upon completing the first data reception, the `HAL_UART_RxCpltCallback()` function was triggered, acting as an interrupt callback to process the received data. At the end of this callback, the `HAL_UART_Receive_DMA()` function was called again, re-enabling the DMA transfer. This cyclical use of the DMA receive function ensured continuous communication without CPU intervention, making the process efficient and suitable for real-time applications.

The GPS module outputs location data in the form of NMEA sentences, which follow a standardized format. The standardized format for GPRMC sentence by NMEA is shown below in Figure 8.

RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data look similar to:

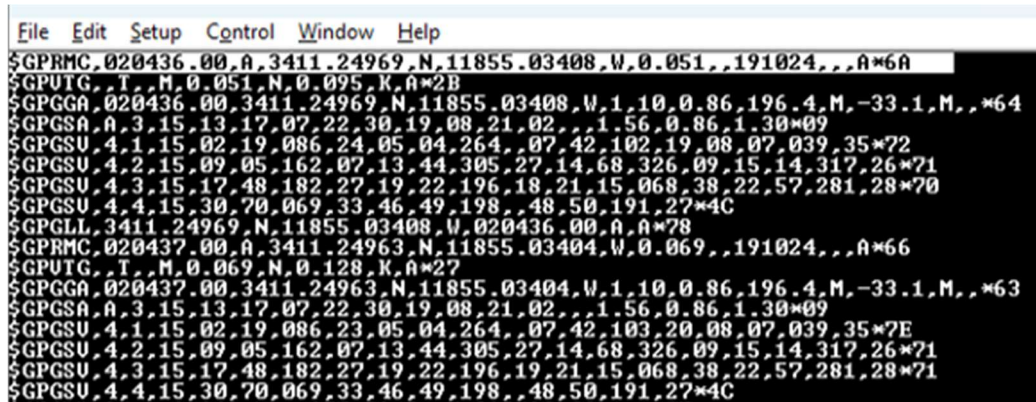
```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

Where:

RMC	Recommended Minimum sentence C
123519	Fix taken at 12:35:19 UTC
A	Status A=active or V=Void.
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
022.4	Speed over the ground in knots
084.4	Track angle in degrees True
230394	Date - 23rd of March 1994
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

Figure 8. GPRMC Sentence Format

The GPRMC sentence (Recommended Minimum Specific GPS/Transit Data) was extracted as it contains essential information, including time, status, latitude, longitude, speed, and date. As shown in the Figure 9, a GPRMC sentence looks like:



```

File Edit Setup Control Window Help
$GPRMC,020436.00,A,3411.24969,N,11855.03408,W,0.051,191024,0.0,A*6A
$GPUTG,T,M,0.051,N,0.095,K,A*2B
$GPGGA,020436.00,3411.24969,N,11855.03408,W,1.10,0.86,196.4,M,-33.1,M,0.0,0.0,0.0
$GPGSA,A,3,15,13,17,07,22,30,19,08,21,02,1.56,0.86,1.30*09
$GPGSU,4,1,15,02,19,086,24,05,04,264,07,42,102,19,08,07,039,35*72
$GPGSU,4,2,15,09,05,162,07,13,44,305,27,14,68,326,09,15,14,317,26*71
$GPGSU,4,3,15,17,48,182,27,19,22,196,18,21,15,068,38,22,57,281,28*70
$GPGSU,4,4,15,30,70,069,33,46,49,198,48,50,191,27*4C
$GPGLL,3411.24969,N,11855.03408,W,020436.00,A,A*78
$GPRMC,020437.00,A,3411.24963,N,11855.03404,W,0.069,191024,0.0,A*66
$GPUTG,T,M,0.069,N,0.128,K,A*27
$GPGGA,020437.00,3411.24963,N,11855.03404,W,1.10,0.86,196.4,M,-33.1,M,0.0,0.0,0.0
$GPGSA,A,3,15,13,17,07,22,30,19,08,21,02,1.56,0.86,1.30*09
$GPGSU,4,1,15,02,19,086,23,05,04,264,07,42,103,20,08,07,039,35*7E
$GPGSU,4,2,15,09,05,162,07,13,44,305,27,14,68,326,09,15,14,317,26*71
$GPGSU,4,3,15,17,48,182,27,19,22,196,19,21,15,068,38,22,57,281,28*71
$GPGSU,4,4,15,30,70,069,33,46,49,198,48,50,191,27*4C

```

Figure 9. Raw Data from GPS Module

Here, the fields are comma-separated, with 020436 representing the time in UTC, A/V for status (Active or Void), 3411.24969, N as latitude, 11855.03408, E as longitude, and so on. The `extractGPRMCData()` function uses this standard format as a reference to parse the required fields. It locates the `$GPRMC` sentence using `strstr()`, then tokenizes it using `strtok()` to extract and process specific data fields.

The extracted time is converted from UTC to PST by adjusting the hours. The latitude and longitude values are parsed into degrees, minutes, and fractional seconds, then formatted appropriately along with direction indicators (N/S for latitude, E/W for longitude). The function also verifies the status field to ensure valid data (A for Active). This process ensures accurate extraction of GPS information from the NMEA sentence, making the system capable of providing reliable current time and location for vehicle tracking and theft detection. Please refer

to Appendix to see the exact code implemented for the function `extractGPRMCData()` Results of using the data parsing function are shown below in Figure 10.

```

COM3 - Tera Term VT
File Edit Setup Control Window Help
Longitude: 12229.02223
Longitude Direction: W
Time: 015105.00
Latitude: 3743.26824
Latitude Direction: N
Longitude: 12229.02189
Longitude Direction: W
Time: 015106.00
Latitude: 3743.26903
Latitude Direction: N
Longitude: 12229.02159
Longitude Direction: W
Time: 015108.00
Latitude: 3743.27061
Latitude Direction: N
Longitude: 12229.02114
Longitude Direction: W
Time: 015109.00
Latitude: 3743.27117
Latitude Direction: N
Longitude: 12229.02109
Longitude Direction: W

```

Figure 10. Extracted Data using `extractGPRMCData()`.

During implementation, two key challenges were encountered. First, the antenna reception issue with the initial GPS Neo-6M module resulted in non-responsiveness of the module. This issue was resolved by switching to the Neo-7M module, which supports SMA antennas, providing improved signal reception and enhanced accessibility. Second, buffer management was a challenge due to the continuous stream of GPS data that was variable in size. To ensure there was no loss of the data, a circular buffer had to be implemented. These solutions ensured smooth GPS communication and reliable extraction of real-time location information.

4G Module Communications

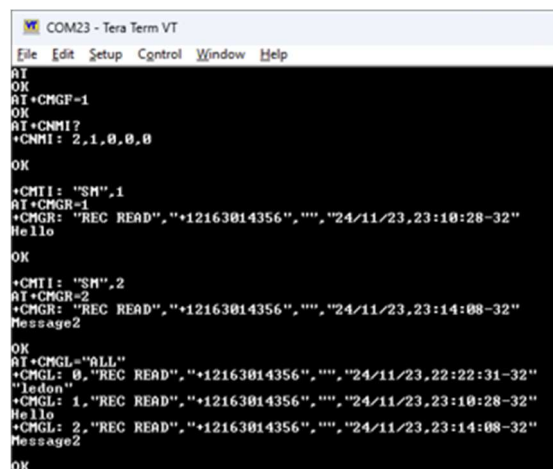
In the 4G Module Communication section, interaction with the SIM7600A-H 4G module was achieved using AT commands sent over UART3. The module was controlled by sending AT commands to configure text messaging, read SMS, and send responses. This was implemented using `HAL_UART_Transmit()` to transmit the AT commands stored in a buffer and `HAL_UART_Receive()` to capture the responses in a separate buffer. For continuous

communication, a check was performed using `HAL_UART_ReceiveAvailable()`, ensuring the UART constantly monitored the incoming data stream.

Key AT commands used in this project included:

- `AT+CMGF=1`: Configures the module to work in Text Mode for SMS communication.
- `AT+CNMI=?`: Enables new message indications so that incoming messages can be processed in real time.
- `AT+CMGR=n`: Reads an SMS at the specified index `n`.
- `AT+CMGS="Mobile-Number"`: Sends a custom "Message" to a specific mobile number.

These commands were first formatted using `sprintf()` into a buffer and sent over UART3 using the `HAL_UART_Transmit()` function. The responses to these commands were stored in the `sms_rx_buffer` for further processing. A flow of commands and response is shown in Figure 11 and Figure 12.

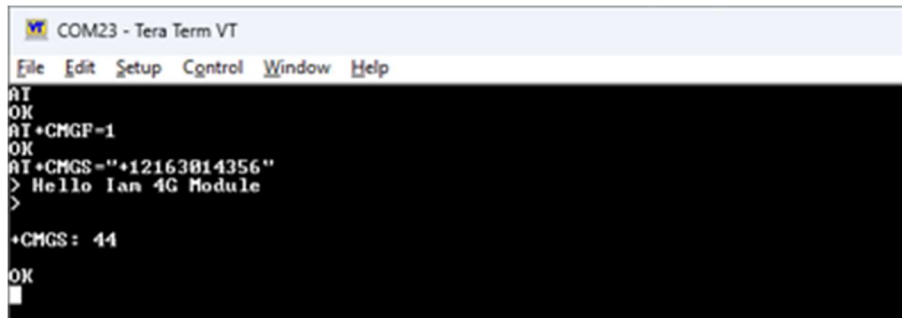


```

COM23 - Tera Term VT
File Edit Setup Control Window Help
AT
OK
AT+CMGF=1
OK
AT+CNMI=?
+CNMI: 2,1,0,0,0
OK
+CMTI: "SH",1
AT+CMGR=1
+CMGR: "REC READ","+12163014356","", "24/11/23,23:10:28-32"
Hello
OK
+CMTI: "SH",2
AT+CMGR=2
+CMGR: "REC READ","+12163014356","", "24/11/23,23:14:08-32"
Message2
OK
AT+CMGL="ALL"
+CMGL: 0,"REC READ","+12163014356","", "24/11/23,22:22:31-32"
"ledon"
+CMGL: 1,"REC READ","+12163014356","", "24/11/23,23:10:28-32"
Hello
+CMGL: 2,"REC READ","+12163014356","", "24/11/23,23:14:08-32"
Message2
OK

```

Figure 11. AT commands to Receive Message



```

COM23 - Tera Term VT
File Edit Setup Control Window Help
AT
OK
AT+CMGF=1
OK
AT+CMGS="+12163814356"
> Hello Ian 4G Module
>
+CMGS: 44
OK

```

Figure 12. AT commands to Send Message Commands

For data processing, functions like `Extract_SMS_Number()` and `Extract_SMS_Message()` were used. The `Extract_SMS_Number()` function identified the index of the new message from the response by extracting the number “n” from the response of the 4G module received when a new message was received. Using this index number “n” the `AT+CMGR=n` command is sent to read the message on index “n” using the `Extract_SMS_Message()` function which further parsed the content of the message. Specific commands like "Get Location", "Lock", "Immobilize", and "Mobilize" were processed to trigger respective actions. For instance, on receiving "Get Location," the system sends the current GPS coordinates using `SendSMS()`. The "Lock" command stored the current location as a reference (parked location), and subsequent unauthorized movement detection sent an alert via SMS. The "Immobilize" and "Mobilize" commands controlled the relay module using GPIO PA5, activating or deactivating the vehicle's ignition circuit.

To ensure smooth operation, the code also handled buffer management by clearing the `sms_rx_buffer` using `memset()` after processing. Additionally, when the SMS index exceeded a threshold, the `AT+CMGD=,4` command was used to delete all messages in the inbox to prevent

overflow. This approach ensured continuous and efficient communication with the 4G module, enabling SMS-based commands, data acquisition, and system control functionalities.

A significant challenge during implementation was the initial use of the SIM900 module, which operates on 2G networks. Since 2G networks are not supported in the USA, the solution was to switch to the SIM7600A-H module, which supports 4G LTE and operates on Band 2 (1900 MHz), ensuring compatibility with US cellular network infrastructure. This upgrade provided reliable connectivity and ensured that the system could send and receive messages seamlessly in the deployment environment. Additionally, buffer management was implemented by clearing `sms_rx_buffer` after processing messages to prevent overwriting issues. This approach enabled efficient and continuous communication with the 4G module, allowing real-time system monitoring and control through SMS.

Anti-Theft Logic Implementation

The system for Smart Anti-Theft Module for vehicles continuously receives GPS data using DMA for efficient, non-blocking updates of the vehicle's location. Simultaneously, in the while loop, the system monitors for new SMS notifications using the blocking mode of `HAL_UART_Receive()`.

When a new SMS is detected, the content is extracted using the `Extract_SMS_Message()` function. If the extracted message contains the command "Lock", the current GPS location is stored in a temporary buffer as the “parked location.” The anti-theft logic then continuously compares this saved location buffer with the continuously updating GPS location buffer. If a discrepancy between the two buffers is detected, it indicates a change in location. Since the “Lock” command was previously issued, this location change is flagged as unauthorized

movement. The system can then take necessary actions, such as sending an SMS alert to the user, notifying them of the unauthorized movement of the vehicle. This logic ensures reliable detection of theft or unauthorized movement while maintaining efficient communication and data monitoring. Detailed code to refer to the implementation logic is mentioned in the Appendix below.

SMS based User Interface

The project employs a simple yet effective communication mechanism using SMS to interact with the system. SMS was chosen as it is lightweight, reliable, and universally compatible with any mobile device that has an active cellular connection. Unlike internet-based communication, SMS does not require a smartphone, dedicated application, or internet connectivity, making it ideal for ensuring accessibility in remote or low-network areas. This approach ensures that users can interact with the system seamlessly, regardless of their device or location.

The SMS interface includes four key commands that allow the user to control and monitor the vehicle:

- "Get Location": This command retrieves the real-time GPS coordinates of the vehicle and sends them back to the user.
- "Immobilize": Triggers the relay to disable the ignition circuit, effectively immobilizing the vehicle.
- "Mobilize": Reverses the immobilization by resetting the relay, restoring the vehicle's functionality.

- "Lock": Saves the vehicle's current location as the “parked location,” and any subsequent change in position triggers an unauthorized movement alert via SMS.

Screenshots of the SMS interface are shown in Figure 13.

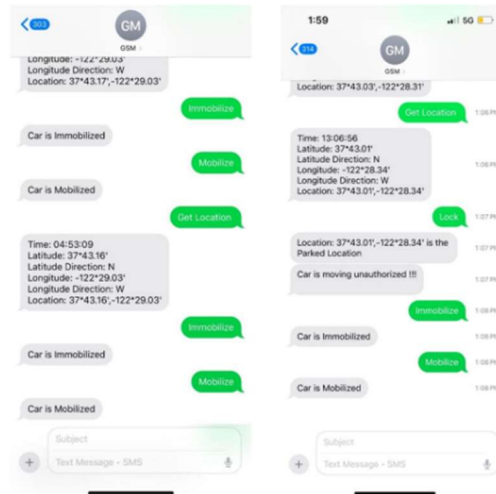


Figure 13. Screenshots of SMS interface.

By using these commands, the SMS interface offers a user-friendly and robust control mechanism for vehicle tracking and theft prevention without requiring complex infrastructure or continuous internet access.

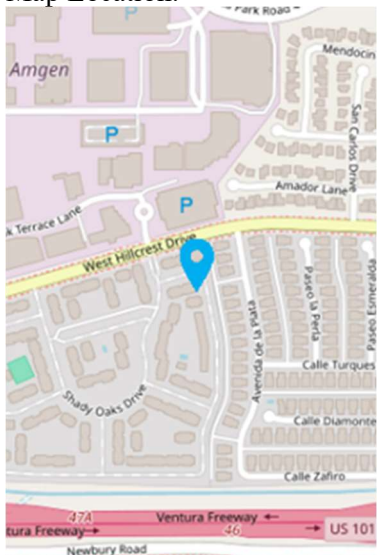
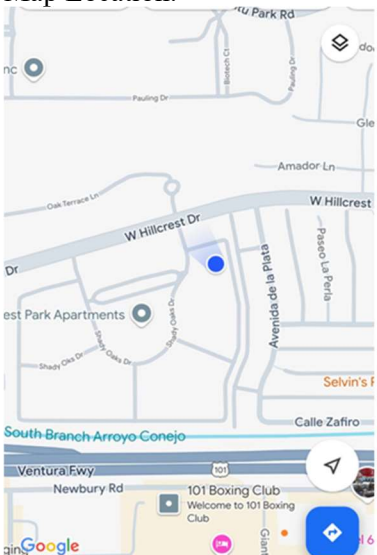
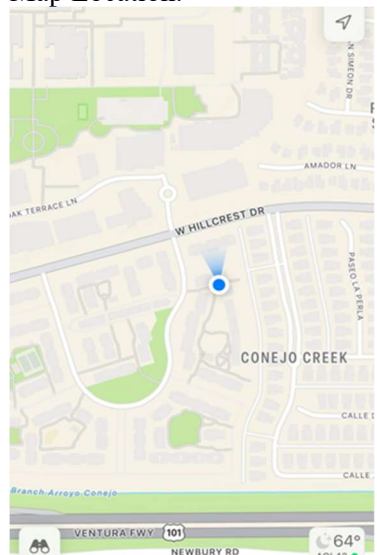
Results

GPS Module Performance Results

The GPS Tracking Performance of the NEO-7M GPS module was evaluated by comparing its output with industry-standard platforms, Google Maps and Apple Maps. The GPS module provided a latitude of 34.18749483, N and longitude of 118.91723467, W, while Google Maps displayed coordinates of 34.1874301, N and 118.9172820, W, and Apple Maps reported

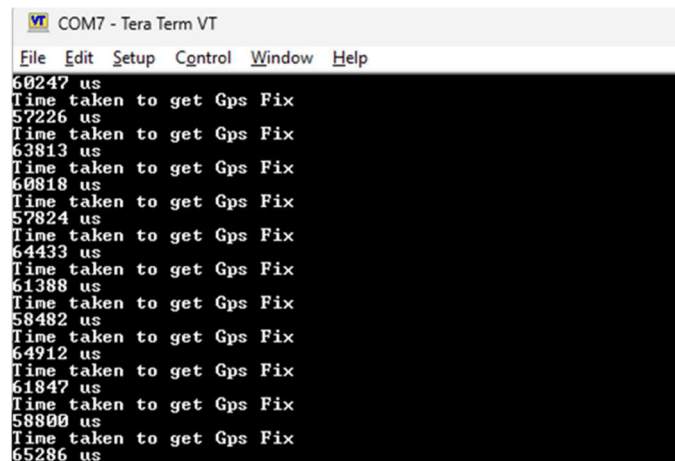
34.18745, N and 118.91723, W. The slight variations between the outputs are within acceptable limits and are attributed to differences in precision, rounding, and real-time signal processing by each platform. The comparison is given in Table 7.

Table 7. Comparison for GPS Tracking Performance

GPS NEO-7M Module	Google Maps	Apple Maps
<p>Latitude :34.1874948,N Longitude :118.9172346,W Map Location:</p>  <p>Details retrieved from NMEA analyzer :</p> <p>https://swairlearn.bluecover.pt/</p>	<p>Latitude :34.1874301,N Longitude:118.9172820,W Map Location:</p>  <p>Dropped pin Near 1710 W Hillcrest Dr, Thousand O... 1 min</p> <p>Directions Start Save</p> <p>Measure distance</p> <p>53PM+X4V Thousand Oaks, California</p> <p>(34.1874831, -118.9172048)</p>	<p>Latitude :34.18745,N Longitude:118.91723,W Map Location:</p>  <p>Details</p> <p>Address 1710 W Hillcrest Dr Newbury Park, CA 91320 United States</p> <p>Coordinates 34.18745° N, 118.91723° W</p>

Despite these minor differences, the GPS module's performance closely aligns with Google Maps and Apple Maps, confirming its reliability for accurate location tracking with a margin error of 2 to 3 meters. This comparison validates the module's effectiveness in providing precise location data, making it suitable for real-world vehicle tracking applications.

To evaluate the responsiveness of the GPS NEO-7M module in acquiring location data, the timer function of the microcontroller used. The timer TIM16 calculated the the time in micro seconds.



```

COM7 - Tera Term VT
File Edit Setup Control Window Help
60247 us
Time taken to get Gps Fix
57226 us
Time taken to get Gps Fix
63813 us
Time taken to get Gps Fix
60818 us
Time taken to get Gps Fix
57824 us
Time taken to get Gps Fix
64433 us
Time taken to get Gps Fix
61388 us
Time taken to get Gps Fix
58482 us
Time taken to get Gps Fix
64912 us
Time taken to get Gps Fix
61847 us
Time taken to get Gps Fix
58800 us
Time taken to get Gps Fix
65286 us

```

Figure 14. Time taken to get GPS Fix

Based on the results displayed in Figure 14, the average time taken by the system to obtain a GPS fix is approximately 61.25 milliseconds, confirming frequent updates to the vehicle's location. During the testing process, the maximum time recorded to achieve a fix was 65.2 milliseconds, as observed in the output. This demonstrates that the GPS module operates efficiently and reliably under test conditions, providing consistent location updates with minimal delay. The ability to update location data approximately every 61.25 milliseconds verifies the

system can track real-time movement with high precision, making it well-suited for vehicle tracking and theft prevention applications.

4G Module Performance Result

The 4G Module Performance was evaluated based on the time taken to read a new message, process it, and send a location response to the user or trigger the actuation of the Relay Module according to the command by the user.

```

Time taken to read message & send location info
53160 us
Time taken to read message & send location info
62299 us
Time taken to read message & send location info
34949 us
Time taken to read message & send location info
31895 us
Time taken to read message & send location info
64646 us
Time taken to read message & send location info
50139 us
Time taken to read message & send location info
16938 us
Time taken to read message & send location info
62092 us

```

Figure 15. Time taken to read a new message and respond to it

From the results, the average time taken for this complete process is approximately 50.55 milliseconds, showcasing the system's responsiveness. The minimum time recorded during the tests was 1.5 milliseconds, highlighting the system's efficiency in handling quick responses under optimal conditions. The maximum time observed was 64.6 milliseconds, which is still well within an acceptable range for real-time communication. This performance demonstrates the system's ability to process incoming messages, extract valid commands, and transmit location information promptly. Combined with the GPS module's update frequency of 61.25 milliseconds, the 4G module effectively ensures minimal delay in relaying critical data to the

user. Such response times make the system reliable for real-world applications where timely communication and location tracking are essential for vehicle monitoring and theft prevention.

Demonstration of Overall Working

The first step of the system's operation is shown in Figure 15. The car is initially static at a particular location. At this point, the user sends the SMS command "Lock" to the system. Upon receiving this message, the system processes it and responds with the current location of the car, which includes "Location: Latitude, Longitude is the Parked Location".

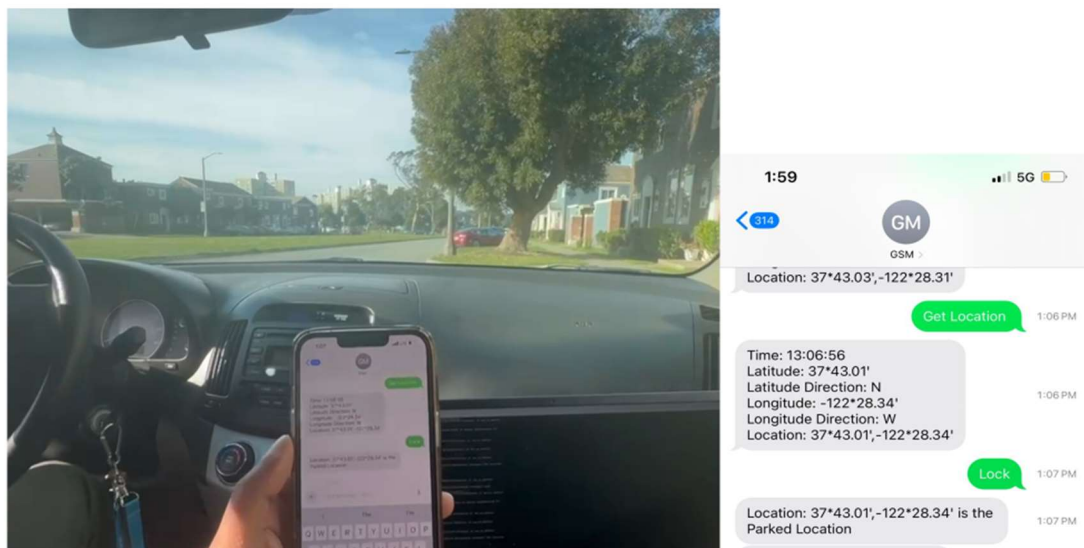


Figure 16. Image taken when car is static and Lock command is sent to system

Simultaneously, the system saves this location as the parked location in its memory for future comparison. This step establishes the baseline reference location, enabling the system to monitor any unauthorized movement of the vehicle.

In the second step of the demonstration, the car begins to move, causing the GPS location to update continuously. The system compares this updated location with the previously saved

parked location. When the two locations do not match, it indicates that the car has moved from its locked position. As a result, the system sends an SMS alert with the message "Car is moving unauthorized!!!" to the user as shown in the Figure 16.

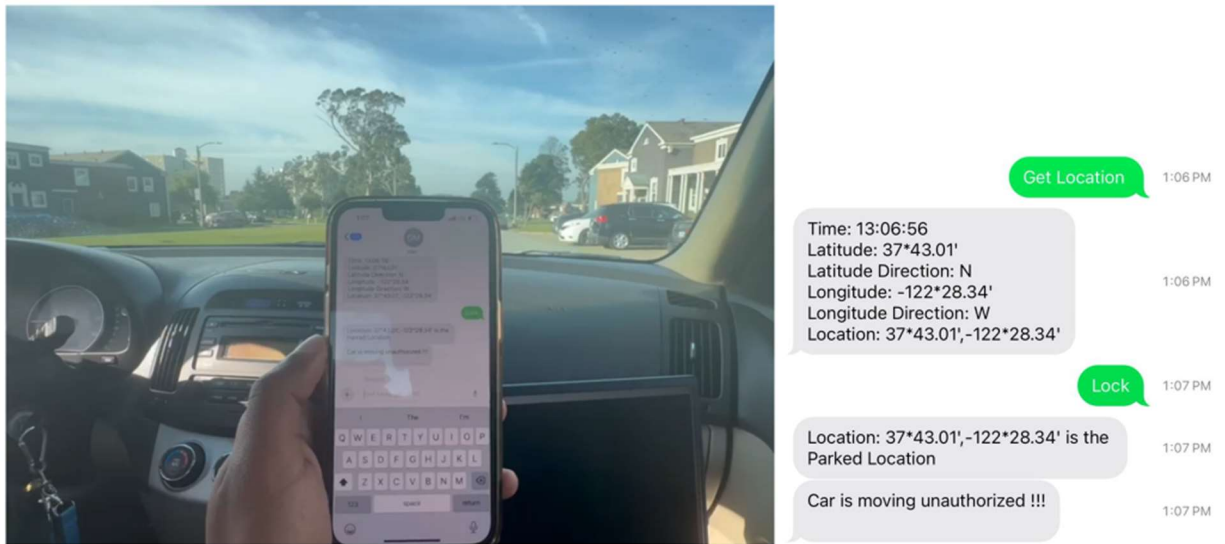


Figure 17. Image taken when car is moving

This notification informs the user of potential unauthorized movement, allowing them to take appropriate action. Based on this alert, the user can decide whether to immobilize the car by sending the "Immobilize" command or request the car's real-time location by sending the "Get Location" command. This step showcases the system's ability to detect theft or unauthorized movement and provides the user with immediate control to secure the vehicle.

Discussion

Future Work

In the future, developing a dedicated mobile application will provide a higher level of user interface, offering a more intuitive and user-friendly experience for vehicle tracking and control. The app will enable features such as real-time GPS location display, vehicle status monitoring, and quick access to system commands like "Lock," "Immobilize," and "Mobilize." While the app will serve as the primary interface, the existing SMS-based control will still function as a fallback mechanism. This ensures that, in cases where the app server is down, internet connectivity is unavailable, or the app fails to work for any reason, users can continue to interact with the system via SMS. This dual-interface approach increases system reliability and accessibility.

Integrating an Onboard Diagnostics (OBD-II) module into the system will enable real-time vehicle health monitoring. The OBD-II module can provide critical information such as engine status, fuel level, battery voltage, error codes, and overall vehicle diagnostics. By analyzing this data, users can receive alerts for issues like engine malfunctions or low fuel levels, helping to address problems before they escalate. This enhancement will not only make the system a theft prevention and tracking solution but also a comprehensive vehicle management tool, improving maintenance and reducing downtime.

Connecting the vehicle tracking system with smart home systems such as Amazon Alexa or Google Home can offer voice-activated control and notifications. For example, users could query the current vehicle location, receive alerts for unauthorized movement, or send commands

like "Lock" or "Immobilize" using simple voice commands. Integration with smart home ecosystems will make the system more convenient and accessible for users, allowing seamless interaction with the vehicle tracking system through their existing smart devices.

Future improvements can also include integrating live traffic data using APIs such as Google Maps or Waze to enhance route optimization. By analyzing real-time traffic conditions, the system can suggest the most efficient routes to the user, saving time and fuel while navigating congested areas. This feature can be particularly beneficial for recovery operations in case of vehicle theft or for general vehicle tracking purposes. Incorporating live traffic updates will transform the system into a dynamic tracking and navigation tool, further enhancing its overall functionality and user experience.

Applications

The vehicle tracking and theft prevention system has a wide range of applications across personal security, fleet management, and rental services. For personal vehicle security, the system enhances safety by providing real-time tracking and unauthorized movement alerts, ensuring vehicle owners are immediately notified of potential theft or misuse. In fleet management, businesses can utilize the system to monitor and manage multiple vehicles in real-time, ensuring delivery trucks or service vehicles remain on course. This enables companies to respond quickly to any theft, route deviations, or delays. Similarly, car rental and leasing services can use the system to monitor vehicle location and usage. For example, rental companies can receive alerts if a vehicle travels outside a designated area or exceeds the agreed rental period, improving security and operational oversight. These applications highlight the system's

versatility in enhancing vehicle safety, optimizing business operations, and offering reliable monitoring solutions.

Conclusion

In conclusion, the project successfully delivers a scalable and effective solution for vehicle security through real-time GPS tracking, remote immobilization, and SMS-based alerts. Challenges, such as antenna reception issues and the lack of 2G compatibility, were overcome by upgrading to the NEO-7M GPS module and the SIM7600A-H 4G module, ensuring reliable operation. The modular design and careful selection of components enabled efficient testing and implementation, resulting in a robust and dependable system. Future enhancements, including advanced vehicle diagnostics, smart home integration, and AI-based predictive analytics, will further improve the system's capabilities and user experience. Overall, the project offers a versatile and practical solution for both personal and commercial vehicle security, addressing key concerns like theft prevention and real-time monitoring.

References

- [1] Saric, I. (2023, July 20). *Theft rates in the USA from 2018 to 2023*. Axios. Retrieved from <https://www.axios.com/>
- [2] Insurance Information Institute. (2022). *Car theft statistics for 2022-2023*. National Insurance Crime Bureau. Retrieved from <https://www.iii.org/>
- [3] Balakrishnan, B., & Murugan, A. V. (2024). Smart Anti-Theft Security System Using IoT. *International Advanced Research Journal in Science, Engineering and Technology*, 11(6), 138-142. DOI: 10.17148/IARJSET.2024.11618
- [4] Bhargavi, B., Padmaja, B. V. K., & Manikanta, K. V. K. S. (2020). Anti-Theft System for Vehicle Security. *International Research Journal of Engineering and Technology*, 7(6), 3867-3870.
- [5] S., Adarsh, N., Abhishek, M., Pranitha, M., & Jaswanth, V. (2022). ARM Based Vehicle Theft Control, Positioning, and Collision Detection System. *International Research Journal of Engineering and Technology*, 9(6), 1439-1442.
- [6] *ARM Based Vehicle Theft Control, Positioning and Collision Detection System*. *International Research Journal of Engineering and Technology (IRJET)*, Vol. 9, Issue 6, June 2022, pp. 1439.
- [7] SIMCom Wireless Solutions. *SIM900 AT Commands Set (SIM900_ATC_V1.00)*. Available at: https://simcom.ee/documents/SIM900/SIM900_AT%20Commands%20Set.pdf

- [8] u-blox AG. NEO-7 u-blox 7 GNSS Modules Data Sheet. Available at: https://content.u-blox.com/sites/default/files/NEO-7_DataSheet_%28UBX-13003830%29.pdf

```

/* USER CODE BEGIN Header */
/**
 *
 *
 * @file : main.c
 * @brief : Main program body
 *
 *
 * @attention
 *
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 *
 */
/* USER CODE END Header */

/* Includes -----*/

#include "main.h"

/* Private includes -----*/

/* USER CODE BEGIN Includes */

#include "string.h"

```



```
#include "stdio.h"
```

```
#include "stdbool.h"
```

```
/* USER CODE END Includes */
```

```
/* Private typedef -----*/
```

```
/* USER CODE BEGIN PTD */
```

```
/* USER CODE END PTD */
```

```
/* Private define -----*/
```

```
/* USER CODE BEGIN PD */
```

```
#define DEBUG_MESSAGE_SIZE 200
```

```
/* GPS declarations -----*/
```

```
#define GPS_RX_BUFFER_SIZE 600
```

```
#define GPS_TX_BUFFER_SIZE 600
```

```
/* SMS declarations -----*/
```

```
#define SMS_RX_BUFFER_SIZE 100
```

```
#define MESSAGE_BUFFER_SIZE 100
```

```
/* USER CODE END PD */
```

```
/* Private macro -----*/
```

```
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */
```

```

/* Private variables -----*/

TIM_HandleTypeDef htim16;

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;
DMA_HandleTypeDef hdma_usart1_rx;
DMA_HandleTypeDef hdma_usart2_tx;

/* USER CODE BEGIN PV */

void extractGPRMCDData(uint8_t *buffer);

int HAL_UART_ReceiveAvailable(UART_HandleTypeDef *huart);
int Extract_SMS_Number(uint8_t *sms_data);
void Clear_RX_Buffer(uint8_t *buffer, size_t size);
bool isArrayZero(const uint8_t *array, size_t size);
int Extract_SMS_Message(uint8_t *sms_data, char *output_message, size_t output_size);
void SendSMS(char *mobileNumber, char *message);

/* USER CODE END PV */

/* Private function prototypes -----*/

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART2_UART_Init(void);

```

```

static void MX_USART1_UART_Init(void);

static void MX_USART3_UART_Init(void);

static void MX_TIM16_Init(void);

/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/

/* USER CODE BEGIN 0 */

char debug_message[DEBUG_MESSAGE_SIZE];

/* GPS declarations -----*/

uint8_t gps_rx_buffer[GPS_RX_BUFFER_SIZE]; // Buffer to store incoming GPS data
uint8_t gps_tx_buffer[GPS_TX_BUFFER_SIZE]; // Buffer to send GPS data to User
uint8_t location_data[300];

char time[15]; // Buffer to store time (hhmmss.sss)
char latitude[20]; // Buffer to store latitude
char longitude[20]; // Buffer to store longitude
char status; // Status of the GPS fix (A or V)
char latitudeDirection[2]; // Lat. Direction
char longitudeDirection[2]; // Lon. Direction
char location_std_format[32]; // Location data in std format
char parked_location[32] = {1};
char temp_location_buffer[32] = {0};

char temp_time[15]; //test buffer to store temporary time for comparison with saved time
char parked_time[15]; //test buffer to store temporary time for comparison with saved time

```

```

/* SMS declarations -----*/

uint8_t sms_rx_buffer[SMS_RX_BUFFER_SIZE]; // Buffer for receiving SMS

char mobileNumber[] = "+12163014356"; // Enter the Mobile Number you want to send to
//char formatted_buffer[SMS_RX_BUFFER_SIZE * 2]; // Buffer to hold the formatted string
char ATcommand[100];
char formatted_buffer[SMS_RX_BUFFER_SIZE * 2]; // Buffer to hold the formatted string

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
/* USER CODE BEGIN 1 */

int sms_number;

char extracted_message[MESSAGE_BUFFER_SIZE];

int message_sent;

int IsParked;

/* USER CODE END 1 */

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */

```

```

SystemClock_Config();

/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_USART2_UART_Init();
MX_USART1_UART_Init();
MX_USART3_UART_Init();
MX_TIM16_Init();
/* USER CODE BEGIN 2 */

//Start Timer
HAL_TIM_Base_Start(&htim16);

//-----
sprintf(ATcommand, "AT+CMGF=1\r\n");
HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 1000);
HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);
HAL_Delay(100);

//-----
sprintf(ATcommand, "AT+CNMI=?\r\n");
HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 1000);
HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);
HAL_Delay(100);

//-----
sprintf(ATcommand, "AT+CMGD=,4\r\n"); // Delete all message in inbox
HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 1000);
HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);

```

```
//-----

HAL_UART_Receive_DMA(&huart1, gps_rx_buffer, GPS_RX_BUFFER_SIZE); // Receive Data from GPS Module
and trigger callback after completed

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
memset(temp_location_buffer, 0, 32 );
message_sent = 0;
strcpy((char *)temp_time, (char *)time); //temp buffer to store time for comparison
strcpy((char *)temp_location_buffer, (char *)location_std_format); //temp buffer to store location for comparison

if(HAL_UART_ReceiveAvailable(&huart3))
{
HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);
sprintf(formatted_buffer, "Contents of sms_rx_buffer: %s\r\n", sms_rx_buffer);
HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);
}

if (isArrayZero(sms_rx_buffer, SMS_RX_BUFFER_SIZE))
{
sprintf(formatted_buffer, "nothing in sms buffer\r\n");
```

```

HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);

}

else

{

    sms_number = Extract_SMS_Number(sms_rx_buffer); // Get sms number from ----- +CMTI: "SM",# ----- if doesn't
exists, sms_number = -1

    if(sms_number != -1)

    {

        sprintf(ATcommand, "AT+CMGR=%d\r\n", sms_number); // Delete all message in inbox
        HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 100);
        HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);
        Extract_SMS_Message(sms_rx_buffer, extracted_message, MESSAGE_BUFFER_SIZE);
        sprintf(formatted_buffer, "extracted messages: %s\r\n", extracted_message);
        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);

        if (strstr(extracted_message, "Get Location") != NULL && !message_sent)

        {

            // Do something for "Get Location"

            SendSMS(mobileNumber, (char *)location_data );

            sprintf(formatted_buffer, "Location Information sent\r\n");
            HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);
            message_sent = 1;

            //timer_val = __HAL_TIM_GET_COUNTER(&htim16) - timer_val;
            //uart_buf_len = sprintf(uart_buf, "%u us \r\n", timer_val);
            //HAL_UART_Transmit(&huart2, (uint8_t *)uart_buf, uart_buf_len, 100);

```

```

    }

    else if (strstr(extracted_message, "Lock") != NULL && !message_sent)
    {
        // Do something for "Lock"

        strcpy((char *)parked_location, (char *)temp_location_buffer); // Store current location in parked location for
comparison with continuously changing location

        strcpy((char *)parked_time, (char *)time);

        strcpy((char *)debug_message, (char *)parked_location); // Store current location in parked location for comparison
with continuously changing location

        strcat((char *)debug_message, " is the Parked Location", sizeof(debug_message) - strlen((char *)debug_message) -
1);

        SendSMS(mobileNumber, debug_message );

        IsParked = 1;

        message_sent = 1;

    }

    else if (strstr(extracted_message, "Immobilize") != NULL && !message_sent)
    {
        // Do something for "Immobilize"

        SendSMS(mobileNumber, "Car is Immobilized" );

        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);

        sprintf(formatted_buffer, "Car is Immobilized\r\n");

        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);

        message_sent = 1;

    }

    else if (strstr(extracted_message, "Mobilize") != NULL && !message_sent)

```



```

{
    // Do something for "Mobilize"

    SendSMS(mobileNumber, "Car is Mobilized" );

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

    sprintf(formatted_buffer, "Car is mobilized\r\n");

    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 100);

    message_sent = 1;
}

else
{
    // Handle unknown message

    sprintf(debug_message, "Unknown Message\r\n");

    HAL_UART_Transmit(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE, 200);
}

if (sms_number >= 5)
{
    sprintf(ATcommand, "AT+CMGD=,4\r\n"); // Delete all message in inbox

    HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 100);

    HAL_UART_Receive(&huart3, sms_rx_buffer, SMS_RX_BUFFER_SIZE, 100);

    sprintf(debug_message, "All Messages Deleted\r\n");

    HAL_UART_Transmit(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE, 100);
}
}

//memset(location_std_format, 0, sizeof(location_std_format)); // clear the message buffer or else same data of the
buffer will be tx'd again and again.

if (IsParked)
{

```

```

int a = strcmp((char *)parked_location, (char *)temp_location_buffer);

if(!a)
{
}

else
{
    sprintf(formatted_buffer, "Car is moving unauthorized !!!\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
    SendSMS(mobileNumber, "Car is moving unauthorized !!!\r\n" );
    IsParked = 0;
}
}

/*
if (IsParked)
{
    int a = strcmp((char *)parked_time, (char *)temp_time);

    if(!a)
    {
        //Parked time and current time is same

        sprintf(formatted_buffer, "Parked Time ==== Current Time\r\n");
        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
        sprintf(formatted_buffer, "Parked Time:\r\n");
        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
        HAL_UART_Transmit(&huart2, (uint8_t *)parked_time, strlen(parked_time), 200);
        sprintf(formatted_buffer, "\r\n");
        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
        sprintf(formatted_buffer, "Current Time:\r\n");
        HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
        HAL_UART_Transmit(&huart2, (uint8_t *)temp_time, strlen(temp_time), 200);
    }
}

```

```

    sprintf(formatted_buffer, "\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);

}

else
{
    //Parked time and current time is different
    sprintf(formatted_buffer, "Parked Time xxxxx Current Time\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
    sprintf(formatted_buffer, "Parked Time:\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
    HAL_UART_Transmit(&huart2, (uint8_t *)parked_time, strlen(parked_time), 200);
    sprintf(formatted_buffer, "\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
    sprintf(formatted_buffer, "Current Time:\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);
    HAL_UART_Transmit(&huart2, (uint8_t *)temp_time, strlen(temp_time), 200);
    sprintf(formatted_buffer, "\r\n");
    HAL_UART_Transmit(&huart2, (uint8_t *)formatted_buffer, strlen(formatted_buffer), 200);

    IsParked = 0;

}

}

*/

memset(extracted_message, 0, MESSAGE_BUFFER_SIZE);

memset(debug_message, 0, DEBUG_MESSAGE_SIZE);

}

/* USER CODE END 3 */

}

```

```

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
     */
    if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 1;
    RCC_OscInitStruct.PLL.PLLN = 10;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;

```

```

if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;

RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;

RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
{
    Error_Handler();
}
}

/**
 * @brief TIM16 Initialization Function
 * @param None
 * @retval None
 */

static void MX_TIM16_Init(void)
{
    /* USER CODE BEGIN TIM16_Init 0 */

    /* USER CODE END TIM16_Init 0 */

    /* USER CODE BEGIN TIM16_Init 1 */

```

```

/* USER CODE END TIM16_Init 1 */

htim16.Instance = TIM16;
htim16.Init.Prescaler = 0;
htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
htim16.Init.Period = 65535;
htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim16.Init.RepetitionCounter = 0;
htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN TIM16_Init 2 */

/* USER CODE END TIM16_Init 2 */

}

/**
 * @brief USART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART1_UART_Init(void)
{
    /* USER CODE BEGIN USART1_Init 0 */

    /* USER CODE END USART1_Init 0 */

    /* USER CODE BEGIN USART1_Init 1 */

    /* USER CODE END USART1_Init 1 */

    huart1.Instance = USART1;

```

```

huart1.Init.BaudRate = 9600;

huart1.Init.WordLength = UART_WORDLENGTH_8B;

huart1.Init.StopBits = UART_STOPBITS_1;

huart1.Init.Parity = UART_PARITY_NONE;

huart1.Init.Mode = UART_MODE_TX_RX;

huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;

huart1.Init.OverSampling = UART_OVERSAMPLING_16;

huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;

huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;

if (HAL_UART_Init(&huart1) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN USART1_Init 2 */

/* USER CODE END USART1_Init 2 */

}

/**
 * @brief USART2 Initialization Function
 *
 * @param None
 *
 * @retval None
 */

static void MX_USART2_UART_Init(void)
{
    /* USER CODE BEGIN USART2_Init 0 */

    /* USER CODE END USART2_Init 0 */

    /* USER CODE BEGIN USART2_Init 1 */

    /* USER CODE END USART2_Init 1 */

    huart2.Instance = USART2;

```

```

huart2.Init.BaudRate = 9600;

huart2.Init.WordLength = UART_WORDLENGTH_8B;

huart2.Init.StopBits = UART_STOPBITS_1;

huart2.Init.Parity = UART_PARITY_NONE;

huart2.Init.Mode = UART_MODE_TX_RX;

huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;

huart2.Init.OverSampling = UART_OVERSAMPLING_16;

huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;

huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;

if (HAL_UART_Init(&huart2) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN USART2_Init 2 */

/* USER CODE END USART2_Init 2 */

}

/**
 * @brief USART3 Initialization Function
 *
 * @param None
 *
 * @retval None
 */

static void MX_USART3_UART_Init(void)
{
    /* USER CODE BEGIN USART3_Init 0 */

    /* USER CODE END USART3_Init 0 */

    /* USER CODE BEGIN USART3_Init 1 */

    /* USER CODE END USART3_Init 1 */

    huart3.Instance = USART3;

```



```

huart3.Init.BaudRate = 115200;

huart3.Init.WordLength = UART_WORDLENGTH_8B;

huart3.Init.StopBits = UART_STOPBITS_1;

huart3.Init.Parity = UART_PARITY_NONE;

huart3.Init.Mode = UART_MODE_TX_RX;

huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;

huart3.Init.OverSampling = UART_OVERSAMPLING_16;

huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;

huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;

if (HAL_UART_Init(&huart3) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN USART3_Init 2 */

/* USER CODE END USART3_Init 2 */

}

/**
 * Enable DMA controller clock
 */

static void MX_DMA_Init(void)
{
    /* DMA controller clock enable */
    __HAL_RCC_DMA1_CLK_ENABLE();

    /* DMA interrupt init */

    /* DMA1_Channel5_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA1_Channel5_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA1_Channel5_IRQn);

    /* DMA1_Channel7_IRQn interrupt configuration */

```

```

HAL_NVIC_SetPriority(DMA1_Channel7_IRQn, 0, 0);

HAL_NVIC_EnableIRQ(DMA1_Channel7_IRQn);

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* USER CODE BEGIN MX_GPIO_Init_1 */

    /* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */

    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

    /*Configure GPIO pin : B1_Pin */
    GPIO_InitStruct.Pin = B1_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;

```

```

HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : PA5 */
GPIO_InitStruct.Pin = GPIO_PIN_5;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // char uart_buf[50];
    // int uart_buf_len;
    // uint16_t timer_val;

    if(huart->Instance==USART1)
    {
        // timer_val = __HAL_TIM_GET_COUNTER(&htim16);

        //printf(debug_message, "Getting Location Data...\r\n");
        //HAL_UART_Transmit(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE, 100);

        memcpy(gps_tx_buffer, gps_rx_buffer, GPS_RX_BUFFER_SIZE);

        // Extract data only if the GPRMC sentence is found

```

```

if (strstr((char *)gps_tx_buffer, "$GPRMC") != NULL)
{
    extractGPRMCDData(gps_tx_buffer);

    // Format the Time, latitude and longitude into a message

    memset(location_data, 0, sizeof(location_data)); // clear the message buffer or else same data of the buffer will be tx'd
again and again.

```

```

    memset(location_std_format, 0, sizeof(location_std_format)); // clear the message buffer or else same data of the
buffer will be tx'd again and again.

```

```

// Construct message with all information

strncat((char *)location_data, "Time: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, time, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, "\n", sizeof(location_data) - strlen((char *)location_data) - 1); // Include \r with \n
strncat((char *)location_data, "Latitude: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, latitude, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, "\n", sizeof(location_data) - strlen((char *)location_data) - 1); // Include \r with \n
strncat((char *)location_data, "Latitude Direction: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, latitudeDirection, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, "\n", sizeof(location_data) - strlen((char *)location_data) - 1); // Include \r with \n
strncat((char *)location_data, "Longitude: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, longitude, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, "\n", sizeof(location_data) - strlen((char *)location_data) - 1); // Include \r with \n
strncat((char *)location_data, "Longitude Direction: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, longitudeDirection, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, "\n", sizeof(location_data) - strlen((char *)location_data) - 1); // Include \r with \n
strncat((char *)location_data, "Location: ", sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, latitude, sizeof(location_data) - strlen((char *)location_data) - 1);
strncat((char *)location_data, ",", sizeof(location_data) - strlen((char *)location_data) - 1); // Include , between lat.
and long.

```

```

strncat((char *)location_data, longitude, sizeof(location_data) - strlen((char *)location_data) - 1);

// Create location in standard format

strncat((char *)location_std_format, "Location: ", sizeof(location_std_format) - strlen((char *)location_std_format) -
1);

strncat((char *)location_std_format, latitude, sizeof(location_std_format) - strlen((char *)location_std_format) - 1);

strncat((char *)location_std_format, ",", sizeof(location_std_format) - strlen((char *)location_std_format) - 1); //
Include , between lat. and long.

strncat((char *)location_std_format, longitude, sizeof(location_std_format) - strlen((char *)location_std_format) - 1);

// Transmit the formatted message using DMA

if (strlen((char *)latitude) != '\0' || strlen((char *)longitude) != '\0')
{
//HAL_UART_Transmit_DMA(&huart2, location_std_format, strlen((char *)location_std_format));

// sprintf(debug_message, "Time taken to get Gps Fix\r\n");

// HAL_UART_Transmit(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE, 100);

// timer_val = __HAL_TIM_GET_COUNTER(&htim16) - timer_val;

// uart_buf_len = sprintf(uart_buf, "%u us \r\n", timer_val);

// HAL_UART_Transmit(&huart2, (uint8_t *)uart_buf, uart_buf_len, 100);

}

else
{

memset(location_data, 0, sizeof(location_data)); // clear the location_data buffer or else same data of the buffer will be
tx'd again and again.

// Construct location_data with all information

sprintf(debug_message, "Invalid Format Location data\r\n");

HAL_UART_Transmit(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE, 100);

}

```

```

}

else
{
    sprintf(debug_message, "Finding GPS fix...\r\n");
    HAL_UART_Transmit_DMA(&huart2, (uint8_t *)debug_message, DEBUG_MESSAGE_SIZE);
}

//Continuously receive from UART.
HAL_UART_Receive_DMA(&huart1, gps_rx_buffer, GPS_RX_BUFFER_SIZE);
}
}

// Function to extract data from the $GPRMC sentence
void extractGPRMCData(uint8_t *buffer) {
    char *sentenceStart = strstr((char *)buffer, "$GPRMC");

    if (sentenceStart != NULL) {
        char *token;

        // Extract the time field (1st field after $GPRMC)
        token = strtok(sentenceStart, ",");
        token = strtok(NULL, ",");
        if (token != NULL) {
            strncpy(time, token, sizeof(time) - 1);
            time[sizeof(time) - 1] = '\0';

            // Convert time from UTC to PST
            int hours = (time[0] - '0') * 10 + (time[1] - '0');
            int minutes = (time[2] - '0') * 10 + (time[3] - '0');
            int seconds = (time[4] - '0') * 10 + (time[5] - '0');
            hours -= 8; // Convert UTC to PST

```

```

if (hours < 0) {

hours += 24;

}

snprintf(time, 15, "%02d:%02d:%02d", hours, minutes, seconds);

}


// Extract the status field

token = strtok(NULL, ",");

if (token != NULL) {

status = token[0];

}


// Extract and convert latitude and longitude if status is 'A'

if (status == 'A') {

// Latitude

token = strtok(NULL, ",");

if (token != NULL) {

strncpy(latitude, token, sizeof(latitude) - 1);

latitude[sizeof(latitude) - 1] = '\0';


// Parse latitude

int degrees = (latitude[0] - '0') * 10 + (latitude[1] - '0');

int minutes = (latitude[2] - '0') * 10 + (latitude[3] - '0');

int fraction = 0;

int fractionLen = 0;

for (int i = 5; latitude[i] != '\0' && latitude[i] != ','; ++i) {

fraction = fraction * 10 + (latitude[i] - '0');

fractionLen++;

}

```

```

// Compute scaling factor manually (10^fractionLen)

int scalingFactor = 1;

for (int i = 0; i < fractionLen; i++) {
    scalingFactor *= 10;
}

int seconds = (fraction * 60) / scalingFactor; // Scale to seconds

token = strtok(NULL, ","); //Latitude Direction

if (token != NULL && token[0] == 'S') {
    snprintf(latitude, 20, "-%d*%02d.%02d", degrees, minutes, seconds);
    strncpy(latitudeDirection, token, sizeof(latitudeDirection) - 1);
}

else {
    snprintf(latitude, 20, "%d*%02d.%02d", degrees, minutes, seconds);
    strncpy(latitudeDirection, token, sizeof(latitudeDirection) - 1);
}

}

// // Latitude Direction

// token = strtok(NULL, ",");

// if (token != NULL) {

//     strncpy(latitudeDirection, token, sizeof(latitudeDirection) - 1);

// }

// Longitude

token = strtok(NULL, ",");

if (token != NULL) {

```



```

strncpy(longitude, token, sizeof(longitude) - 1);

longitude[sizeof(longitude) - 1] = '\0';

// Parse longitude

int degrees = (longitude[0] - '0') * 100 + (longitude[1] - '0') * 10 + (longitude[2] - '0');

int minutes = (longitude[3] - '0') * 10 + (longitude[4] - '0');

int fraction = 0;

int fractionLen = 0;

for (int i = 6; longitude[i] != '\0' && longitude[i] != ';'; ++i) {

    fraction = fraction * 10 + (longitude[i] - '0');

    fractionLen++;

}

// Compute scaling factor manually (10^fractionLen)

int scalingFactor = 1;

for (int i = 0; i < fractionLen; i++) {

    scalingFactor *= 10;

}

int seconds = (fraction * 60) / scalingFactor; // Scale to seconds

token = strtok(NULL, ","); // Longitude direction (E/W)

if (token != NULL && token[0] == 'W') {

    snprintf(longitude, 20, "-%d*%02d.%02d", degrees, minutes, seconds);

    strncpy(longitudeDirection, token, sizeof(longitudeDirection) - 1);

}

else {

    snprintf(longitude, 20, "%d*%02d.%02d", degrees, minutes, seconds);

    strncpy(longitudeDirection, token, sizeof(longitudeDirection) - 1);

```

```

}
}

// Longitude Direction
// token = strtok(NULL, ",");
// if (token != NULL) {
//   strncpy(longitudeDirection, token, sizeof(longitudeDirection) - 1);
// }
}
}
}

int HAL_UART_ReceiveAvailable(UART_HandleTypeDef *huart)
{
// Check if the UART's receive data register is not empty
if (__HAL_UART_GET_FLAG(huart, UART_FLAG_RXNE)) {
// Clear the RXNE flag (optional as it is cleared when reading data)
return 1; // Data available to read
}
return 0; // No data available
}

int Extract_SMS_Number(uint8_t *buffer)
{
// Convert buffer to null-terminated string
const char *data_str = (const char *)buffer;
// Locate the "+CMTI:" string in the buffer
const char *start = strstr(data_str, "+CMTI:");
if (start == NULL) {

```

```

return -1; // "+CMTI:" not found

}

// Locate the "SM" part within "+CMTI:"
start = strstr(start, "\"SM\"");

if (start == NULL) {

return -1; // "\"SM\"," not found

}

// Move past "\"SM\","
start += strlen("\"SM\"");

// Extract the number after "\"SM\","

int sms_number = -1;

if (sscanf(start, "%d", &sms_number) == 1) {

// Validate the extracted number is within the valid range (0-29)

if (sms_number >= 0 && sms_number <= 29) {

return sms_number; // Valid SMS number

}

}

return -1; // Return error if parsing fails or number is out of range

}

bool isArrayZero(const uint8_t *array, size_t size) {

for (size_t i = 0; i < size; i++) {

if (array[i] != 0) {

return false; // Found a non-zero element

}

}

return true; // All elements are zero

}

```

```

int Extract_SMS_Message(uint8_t *sms_data, char *output_message, size_t output_size)
{
    // Convert the uint8_t buffer to a null-terminated string

    char *data_str = (char *)sms_data;

    // Locate the first instance of "\r\n" to skip the header

    char *start = strstr(data_str, "\r\n");

    if (start == NULL) {
        // If "\r\n" not found, return failure

        return 0;
    }

    // Move to the start of the actual message (skip "\r\n")

    start += 2;

    // Locate the next "\r\n" which marks the end of the message

    char *end = strstr(start, "\r\n");

    if (end == NULL) {
        // If the end "\r\n" not found, return failure

        return 0;
    }

    end += 2;

    char *final_end = strstr(end, "\r\n");

    if (final_end == NULL) {
        // If the end "\r\n" not found, return failure

        return 0;
    }

    // Calculate the message length

    size_t message_length = final_end - end;

    if (message_length >= output_size) {

```

```

// Truncate if message exceeds the output buffer size

message_length = output_size - 1;

}

// Copy the message content to the output buffer and null-terminate it
strncpy(output_message, end, message_length);

output_message[message_length] = '\0';

return 1; // Return success

}

void SendSMS(char *mobileNumber, char *message) {

sprintf(ATcommand, "AT+CMGS=\"%s\"\r\n", mobileNumber);

HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 100);

HAL_Delay(100);

HAL_UART_Transmit(&huart3, (uint8_t *)message, strlen(message), 100);

sprintf(ATcommand, "%c\r\n", 0x1a); // Ctrl+Z

HAL_UART_Transmit(&huart3, (uint8_t *)ATcommand, strlen(ATcommand), 100);

}

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 *
 * @retval None
 */

void Error_Handler(void)

{

/* USER CODE BEGIN Error_Handler_Debug */

/* User can add his own implementation to report the HAL error return state */

__disable_irq();

while (1)

{

```

```

}

/* USER CODE END Error_Handler_Debug */

}

#ifdef USE_FULL_ASSERT

/**
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */

void assert_failed(uint8_t *file, uint32_t line)
{
/* USER CODE BEGIN 6 */

/* User can add his own implementation to report the file name and line number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

/* USER CODE END 6 */

}

#endif /* USE_FULL_ASSERT */

```