GPS Tracking System Using U-Blox GPS and SIM800L GSM Modules for Current Location Sharing

Anas M. Wagih [a, 1] , Marwan T. Ghazy [b]

[a] *Department of Electronics and Communications Engineering, Arab Academy for Science, Technology & Maritime Transport, Smart Village, EG*

[b] *Department of Electronics and Communications Engineering, Arab Academy for Science, Technology & Maritime Transport, Smart Village, EG*

Abstract

Global Positioning System (GPS) based tracking systems have been widely integrated into many industries, including transportation, supply chain and logistics—cargo transportation, aviation and maritime navigation, consumer products for personal tracking purposes, and wildlife & environmental monitoring—migration patterns. This paper presents a real-life implementation of a low-power GPS tracker with current location sharing via Global System for Mobile Telecommunications (GSM), utilizing the U-Blox GPS module and SIM800L GSM module. The system acquires precise geolocational data and transmits it through GSM using the GPRS protocol, enabling current location access on map services. We discuss hardware and software implementation, and the communication protocols used. Experimental results demonstrate a reliable connection in different dynamic environments.

Keywords: Global Positioning System; GSM; Live Location Sharing; Low-Power

1. Introduction

Existing GPS-based tracking solutions typically use either Wi-Fi, cellular networks, or satellite communication to transmit location data. Regardless, many consumer-grade systems require high-cost infrastructure. SMS-based systems, being simple, provide on-demand location updates rather than continuous updates. Which, for most cases would be sufficient. It is also computationally less demanding and is low-power.

This report presents the development of a low-power GPS tracking system capable of current location sharing using GSM communication. The system integrates the U-Blox GPS module for geolocation acquisition and the SIM800L GSM module for data transmission. Implementing SMS-based communication between the tracker and the user for a lower-cost and lesser complexity – limited features.

[1] Corresponding author.
Email address: anaswagih35@gmail.com (Anas M. Wagih)

We describe the hardware and software architecture of the system, including the communication protocols used for reliable data transmission. A performance evaluation is conducted to assess power efficiency, data transmission reliability, and overall system accuracy under different conditions. The results demonstrate the feasibility of an affordable and energy-efficient GSM-based tracking solution for practical deployment.

2. Hardware Architecture of U-Blox GPS Module

In this section, we briefly review the hardware architecture of the U-Blox (Neo-6) GPS Module given: Internal Components, explaining the flow of data between components, Hardware Features, which shows the available capabilities of the device, Connectivity & Interfaces, presenting the operable communication protocols, and lastly, Power and Performance Considerations, discussing power modes and efficiency.

2.1. Internal components and means of operation

This Demonstration – Using a block diagram shows the data/power flow inside the system.

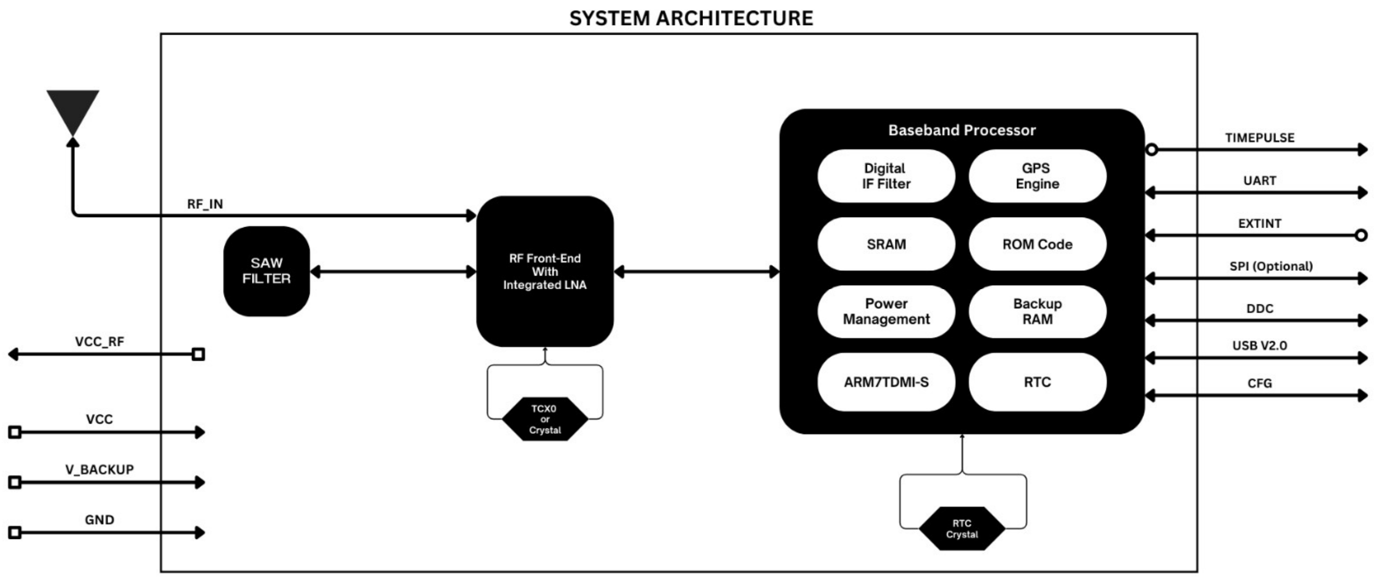


Figure 1: The Block Diagram for the operational flow of signals for the "Neo-6" version of the U-Blox GPS module.

As demonstrated in Figure 1 there are some main active components that the main system depends on to be operational [1], those parts being:

- **U-Blox 6 Positioning Engine** – The core GPS processing unit that computes location.
- **Low-Noise Amplifier (LNA)** – Enhances weak satellite signals for improved reception.
- **SAW Filter (Surface Acoustic Wave)** – Reduces interference by filtering unwanted signals.
- **Temperature-Compensated Crystal Oscillator (TCXO) / Crystal Oscillator** – Ensures stable clock signals for accurate satellite signal reception.
- **Flash Memory / ROM** – Stores firmware and settings.
- **Power Management Unit (PMU)** – Regulates power consumption efficiently.

2.1.1. U-Blox 6 Positioning Engine

For this section, we use the Newton-Raphson method for GPS trilateration, where we are solving a system of **nonlinear equations** to find the receiver's position [4,5]. Therefore, a function should be defined to solve for the Cartesian coordinates:

$$\boldsymbol{f_i(x, y, z, \delta t) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + c \cdot \delta t - p_i} \qquad \textbf{(1)}$$

Each satellite provides a **pseudorange** measurement p_i which is used to estimate the receiver's 3D position (x_i,y_i,z_i):

$$\boldsymbol{p_{n+1} = p_n - J^{-1} \cdot r} \qquad \textbf{(2)}$$

The Jacobian Matrix is then resolved – the partial derivatives with respect to the Cartesian coordinates (x_i,y_i,z_i) as:

$$\boldsymbol{J_i = \left[\frac{\partial f_i}{\partial x}, \frac{\partial f_i}{\partial y}, \frac{\partial f_i}{\partial z}, \frac{\partial f_i}{\partial \delta t}\right]} \qquad \textbf{(3)}$$

$$\boldsymbol{\frac{\partial f_i}{\partial x} = \frac{x-x_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}} \qquad \textbf{(4)}$$

$$\boldsymbol{\frac{\partial f_i}{\partial y} = \frac{y-y_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}} \qquad \textbf{(5)}$$

$$\boldsymbol{\frac{\partial f_i}{\partial z} = \frac{z-z_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}} \qquad \textbf{(6)}$$

$$\boldsymbol{\frac{\partial f_i}{\partial \delta t} = c} \qquad \textbf{(7)}$$

We then perform residual computation, to obtain r_i:

$$\boldsymbol{r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + c \cdot \delta t - p_i} \qquad \textbf{(8)}$$

For the position estimation the Newton-Raphson iteration formula is used with respect to r_i, until convergence condition is reached:

$$\boldsymbol{|r| < \varepsilon} \qquad \textbf{(9)}$$

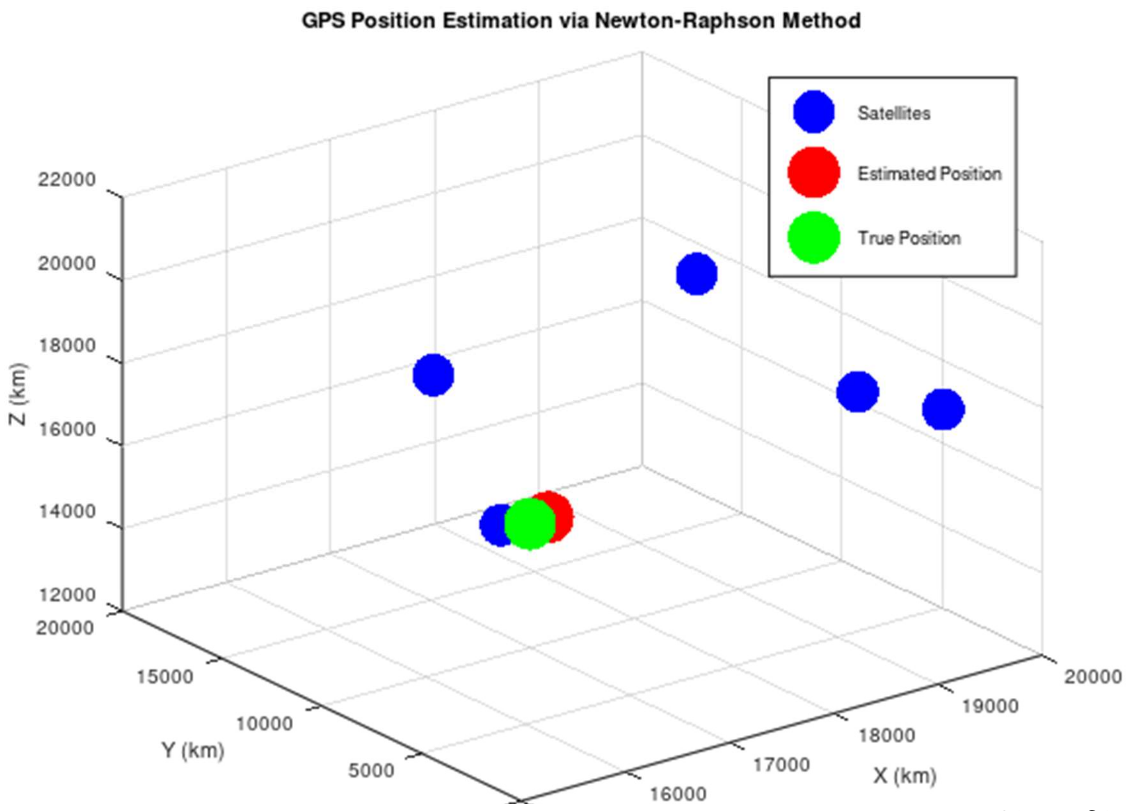


Figure 2: Simulation for the GPS position estimation process using trilateration – function is defined as **f_i(x,y,z, δt)**, using Newton-Raphson's numerical method for refined positional estimation and convergence - with added Gaussian noise for realistic modeling.

2.1.2. Low-Noise Amplifier (LNA)

The Low-Noise Amplifier **(LNA)** in the U-Blox GPS module amplifies weak signals while working on minimizing noise to maintain a high signal-to-noise ratio (SNR) [1, 9], this ensures proper impedance matching, filters out unwanted interference, and enhances the received GPS signal before passing it to the RF front-end for further processing. This improves the module's ability to establish connection and track satellites, even in weak signal environments.

The Low-Noise Amplifier in the U-Blox GPS Module works in a series of steps, the amplification gain factor is first calculated, then the Noise Figure (NF) is worked out, which represents how much noise the LNA introduces, and lastly, the output power is computed.

- **Gain (G)**: Typically **15-20 dB**, is calculated as:

$$\boldsymbol{G_{\mathrm{dB}} = 10\log_{10}\left(\frac{P_{\mathrm{out}}}{P_{\mathrm{in}}}\right)} \qquad \textbf{(10)}$$

- **Noise Figure (NF) Calculation,** measuring how much additional noise the LNA introduces:

$$\boldsymbol{NF = 10\log_{10}\left(\frac{SNR_{\mathrm{in}}}{SNR_{\mathrm{out}}}\right)} \qquad \textbf{(11)}$$

- **Output Power Calculation,** showing how much the Low-Noise Amplifier (LNA) boosts the received GPS signal power, ensuring that the signal is strong enough for further processing:

$$\boldsymbol{P_{\mathrm{out}} = P_{\mathrm{in}} + G} \qquad \textbf{(12)}$$

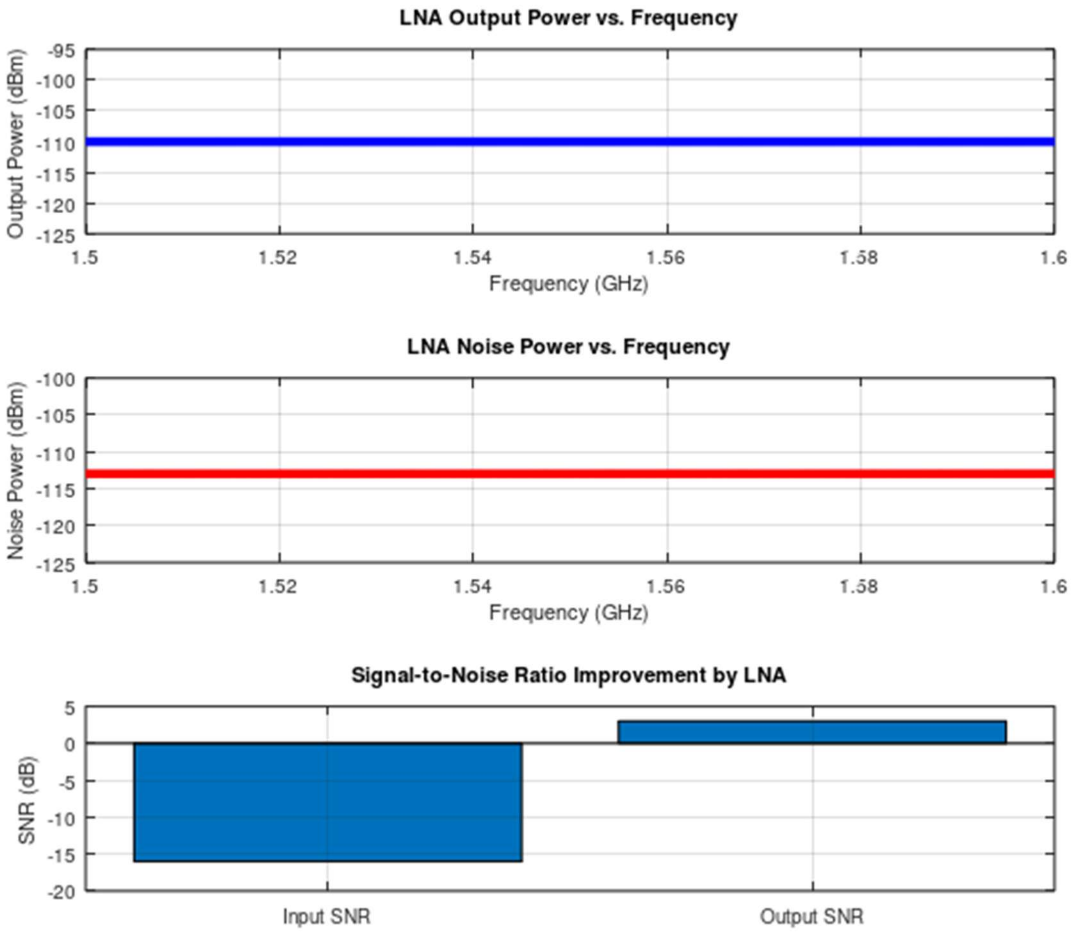


Figure 3: This simulation demonstrates the impact of a Low-Noise Amplifier (LNA) on a weak GPS signal by calculating the signal-to-noise ratio (SNR) before and after amplification. The simulation begins with an **input SNR of -15 dB**, which indicates a weak GPS signal relative to the noise detected; LNA provides a **20 dB gain** to the signal while introducing a small amount of noise (≈ 1 dB – Noise Figure). After the LNA amplification, the **output SNR increases to +5 db**. This means the signal is stronger than the noise — resulting in a much clearer signal that is highly detectable and easily processed by the GPS receiver.

2.1.3. SAW Filter (Surface Acoustic Wave)

The SAW Filter found in the NEO-6 U-Blox GPS Module serves as an electronic filter that uses acoustic waves traveling along the surface of a material to filter out unwanted frequencies, providing high selectivity and low insertion loss [10].

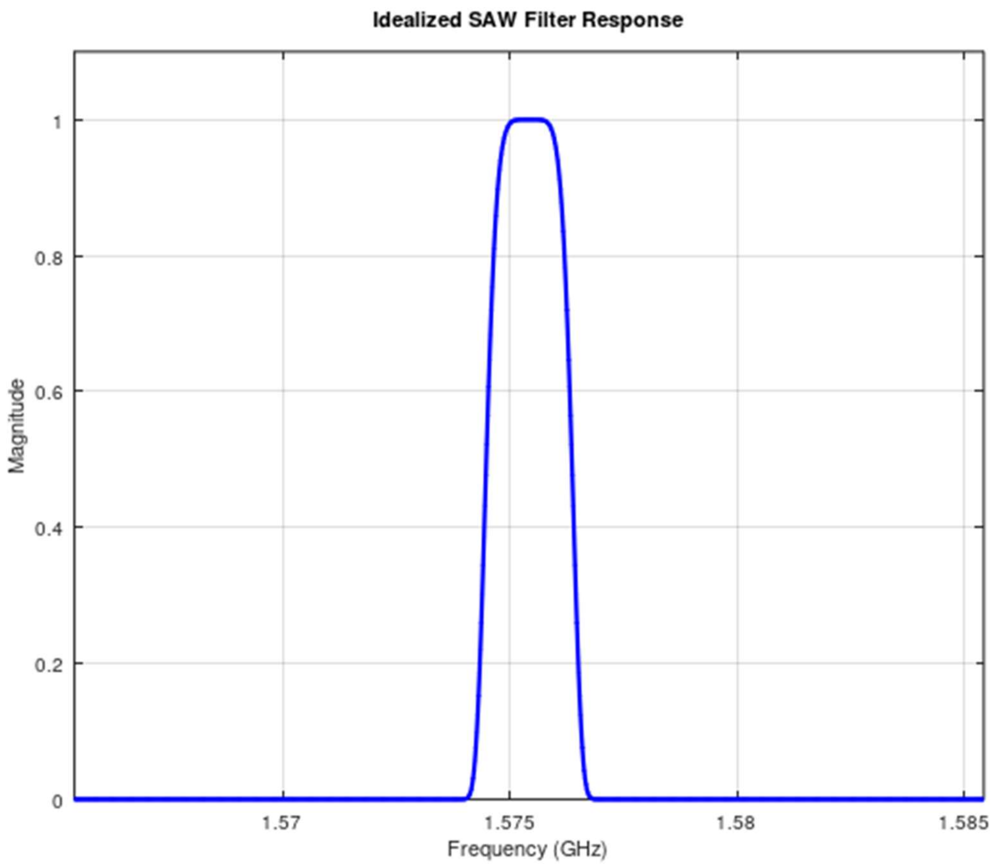


Figure 4: This simulation shows the precision of the SAW Filter response to acoustic waves, showing the sharp passband around the L1 Frequency, peaking at 1.575(GHz), also, showcasing the stopband for signals far away from the L1 frequency band, the filter discards the signals by converting them into acoustic waves that are either absorbed or reflected in a way that prevents them from passing through.

As shown in Figure 4. the SAW Filter works by isolating the GPS signal – the L1 operation frequency (approx. 1.575 GHz) from other radio signals with minimal attenuation, while simultaneously rejecting unwanted frequencies. Therefore, separating the signal from background noise.

2.1.4. Temperature-Compensated Crystal Oscillator (TCXO)

A Temperature-Compensated Crystal Oscillator (TCXO) is a type of timing oscillator that compensates for temperature-induced frequency differences, ensuring a stable and accurate frequency output across a large range of operational temperatures [1]. It is usually used in applications that need precise timing and frequency stability, such as our NEO-6 GPS module.

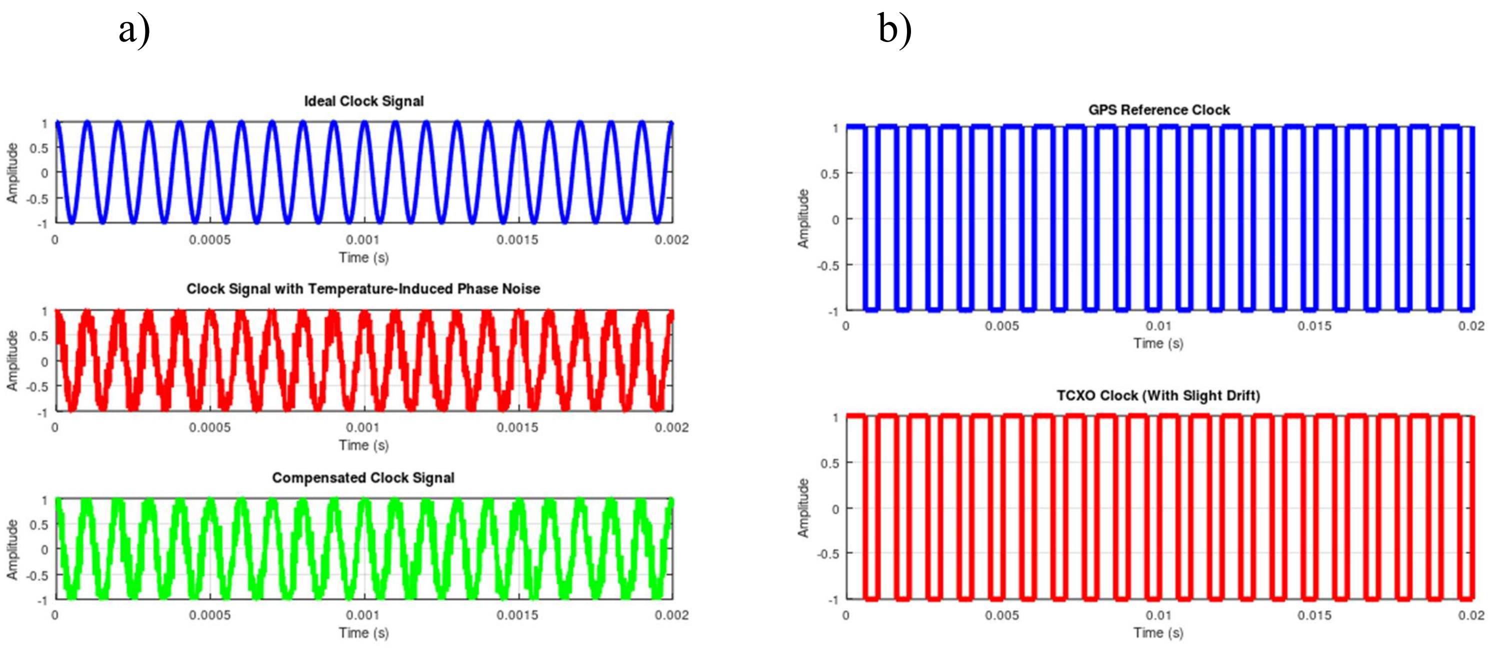


Figure 5: (a) **Temperature-induced Phase Noise** refers to the natural frequency drift that occurs in traditional **crystal oscillators** when the temperature changes, *Eq.13* (Alpha/α – Quartz Crystal temperature coefficient). **TCXO** compensates for this temperature-induced frequency drift, by continuously adjusting the oscillator's frequency to counteract the effects of temperature changes. (b) Shows the GPS reference clock (1 kHz), which is perfectly periodic vs. the **TCXO** clock (1.002 kHz), showcasing slight drift, which is done through VCXO – *Eq.14*, The TCXO works on synchronization through the doppler shift equation *Eq.16*.

There are some important contributing factors to the TXC0 operation in regulating and synchronizing frequency bands in different environments [4,9], those being:

- **Frequency Drift**: A quartz crystal's frequency deviates due to temperature(which can be approximated with a parabolic function), TCXO compensates for this drift by adjusting the capacitance of a **varactor diode** in the oscillator circuit, effectively tuning the resonance frequency.

$$\Delta f(T) = f_0 \cdot \alpha (T - T_0)^2 \qquad \textbf{(13)}$$

- **Compensation via Voltage-Controlled Oscillator (VCO)**: A TCXO typically uses a voltage-controlled Crystal Oscillator (VCXO) to fine-tune the frequency, it generates "Vc" based on a temperature sensor and compensation algorithm (Compensation Lookup Table) stored in the oscillator's memory.

$$\Delta f = K_v \cdot V_c \qquad \textbf{(14)}$$

- **Phase Noise and Allan Deviation:** For GPS applications, frequency stability is crucial, and we analyze it using *Allan deviation*; where y(t) is the normalized frequency deviation, and **τ** is the observation period. TCXO minimizes the short-term frequency noise, improving GPS signal detection and tracking.

$$\sigma_y(\tau) = \sqrt{\frac{1}{2}\langle[y(t+\tau) - y(t)]^2\rangle} \quad \textbf{(15)}$$

- **Clock Synchronization:** The GPS receiver uses the *Doppler shift equation* to track satellites, A stable TCXO ensures minimal frequency error, allowing precise timing synchronization essential for accurate GPS positioning.

$$f_r = f_t\left(\frac{c+v}{c}\right) \quad \textbf{(16)}$$

2.1.5. Flash Memory / ROM

In the Neo-6 GPS module, the flash memory (ROM) plays an important role in storing the firmware that controls the GPS' functionality. This type of memory is non-volatile, meaning it retains the data even when the power is off. It specifically stores [1]:

- GPS Initialization Data: It stores the software for initializing the GPS receiver and communication protocols.
- Configuration Settings: Stores default configurations and calibration data for the module.
- Firmware Updates: the flash memory can also be used for firmware upgrades, allowing the module to receive updates for improved performance or new features are available (if such option is viable).
- Ephemeris Data: While the GPS module often downloads new satellite data from the GPS satellites, some modules store previously received ephemeris data temporarily in flash memory for faster startup (saves computational power) when it first receives power.

This allows the GPS module to function properly and with minimal delay after being powered on.

2.1.6. Power Management Unit (PMU)

The PMU in the Neo-6 GPS module stands for Power Management Unit and is responsible for managing power consumption across different parts of the chip. It handles switching between power modes, such as maximum performance mode, which offers high accuracy but consumes more power, and power-saving mode, which reduces the update rate to conserve energy [1]. The PMU also controls the startup and shutdown of internal modules like the RF and digital core, and monitors voltage levels to protect against undervoltage. Overall, the PMU helps the NEO-6M balance performance and power efficiency, which is particularly important in battery-powered GPS applications.

3. Hardware Architecture of the SIM800L GSM Module

In this section, we will review the hardware architecture of the SIM800L GSM Module, given: Baseband Processor (MCU), explaining the processes of operation inside the module. Radio Frequency (RF), which is responsible for the modulation and demodulation circuits [2]. Power Management and Interface, responsible for power and performance considerations, discussing power modes and efficiency.

3.1. Baseband Processor (MCU)

- **GSM Protocol Stack:**

The baseband processor in the SIM800L manages the GSM protocol stack, which is a layered communication model that defines how data is transmitted between the GSM module and the mobile network; including the Application Layer, SMS Protocol, and Radio Resource Layer.

- **AT Command Set**:

The baseband processor in the SIM800L manages the GSM protocol stack, which is a layered communication model that defines how data is transmitted between the GSM module and the mobile network; including the Application Layer, SMS Protocol, and Radio Resource Layer [7].

- AT+CMGF=1: Sets the SMS message format to text mode.
- AT+CMGS="<Phone Number>": Sends an SMS to the specified phone number.
- AT+CSQ: Checks signal strength.
- AT+CGPSINF=0: Retrieves the GPS position information from the NEO-6M GPS module.
- AT+GETLOC=1: Sends back “Google Maps” current location link.

- **Data Encoding/Decoding**

  The baseband processor ensures that the message content is encoded in a format suitable for GSM transmission. For SMS, the text is typically encoded in **7-bit GSM encoding** [2], which allows a maximum of 160 characters per SMS message. If the message exceeds 160 characters, it will be split into multiple messages.

- **Error Handling and Retransmission**:

The SIM800L module's baseband processor also ensures **error detection and correction** [10] during the transmission of SMS messages. If an SMS fails to send, the baseband processor will retry sending the message until it is successfully delivered or the maximum number of retries is reached.

3.2. Radio Frequency (RF)

This section includes the radio transceiver, and all components required for GSM communication, such as modulation and demodulation circuits. It ensures the signal is properly transmitted and received over the GSM network, using frequency bands like 850 MHz, 900 MHz, 1800 MHz, or 1900 MHz, depending on the region [4,7].

One could perform **frequency analysis** of the RF signal or simulate the **signal strength** for different distances and environments. This can be related to the concept of **free-space path loss** in radio communication.

$$L_f = 20\log_{10}\left(\frac{4\pi d f}{c}\right) \tag{17}$$

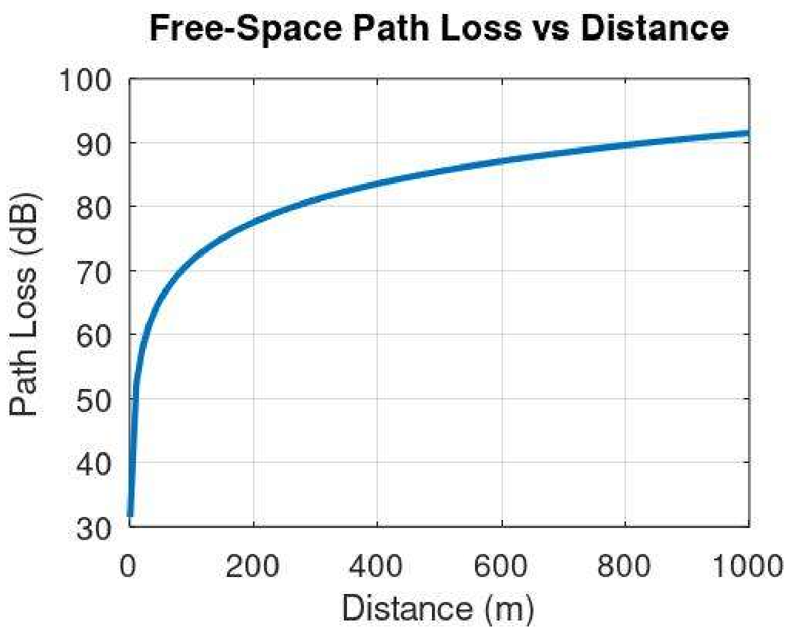


Figure (6): This simulation can help you understand how the path loss varies with distance and frequency, based on the **Free-Space Path Loss** equation; Eq.17.

3.3. Power Management and Interface

This section includes the power supply circuits that regulate voltage levels and manage power distribution within the module. It also includes communication interfaces (like UART) that allow the GSM module to interact with external microcontrollers or processors for data transfer.

- **Calculation/Simulation**: One could calculate voltage regulation efficiency or analyze the current spikes during transmission, as the module typically consumes a high peak current when transmitting.
- **Current Spike Calculation**: If you have the peak current data (`I_peak`), you could calculate the energy consumed during a transmission burst using.

$$E = I_{peak} \cdot V \cdot t \tag{18}$$

4. Hardware and Software - GPS Tracker

This GPS tracker system is built using an Arduino Nano microcontroller as the central processing unit, interfacing with several peripheral modules to provide location tracking, data logging, and communication capabilities [1-3]. The key hardware components include:

4.1. Hardware Components

HEAD TO APPENDIX (A) for FULL CIRCUIT SCHEMATICS

- **Neo-6M GPS Module**: Responsible for acquiring real-time geographical coordinates (latitude, longitude) using signals from GPS satellites.
- **SIM800L GSM Module**: Enables remote transmission of GPS data via SMS or mobile data networks. It acts as the communication backbone of the system.
- **Adafruit 1.14" TFT Display (ST7789)**: A color display used to visualize system status, coordinates, or messages in real-time [8].
- **Arduino Nano V3**: Used for the main processing unit for peripheral control.
- **3.7V 1500mAh LiPo Battery**: Supplies power to the system. It is selected for its lightweight and high energy density, suitable for mobile applications.

4.2. Software Implementation

HEAD TO APPENDIX (B) For FULL SYSTEM CODE

The software for this project is developed using the **Arduino IDE** and leverages several open-source libraries [5,8] to handle the functionality of each module. The system is programmed to:

- **Read GPS data** - from the Neo-6M module using the TinyGPS++ library.
- **Display data -** on the TFT screen using the Adafruit GFX and ST7789 libraries.
- **Send location data -** via the GSM module using AT commands through SoftwareSerial.
- **Monitor power and system status -** handle possible communication errors or GPS signal loss.

The code for the system needs a variety of libraries for proper configuration, shown as follows:

1. Adafruit_GFX.h

- **Function:**
  This is the **core graphics library** from Adafruit.
- **Purpose:**
  It provides basic drawing functions like drawing text, shapes, setting colors, and positioning for displays.
- **Used for:**
  Rendering and formatting text and visuals on the TFT screen.

---

2. Adafruit_ST7789.h

- **Function:**
  This is the **hardware-specific driver** for the ST7789 TFT screen (1.14", 240x135).
- **Purpose:**
  It allows communication with the TFT display over SPI and controls screen-specific features like screen size, color modes, and initialization.
- **Used for:**
  Initializing and drawing on the ST7789 TFT screen.

---

3. SPI.h

- **Function:**
  This is the standard **Serial Peripheral Interface (SPI)** library provided by Arduino.
- **Purpose:**
  Allows the Arduino to communicate with SPI devices like the TFT screen.
- **Used for:**
  Handling SPI communication between Arduino and the TFT display.

---

4. SoftwareSerial.h

- **Function:**
  Enables the use of **multiple serial ports** using software-defined TX and RX pins.
- **Purpose:**
  Because Arduino UNO/Nano only has one hardware serial port, this allows you to create additional serial interfaces (e.g., for GPS or GSM modules).
- **Used for:**
  - gpsSerial on pins 4 (RX) and 3 (TX) for GPS data.
  - gsmSerial on pins 7 (TX) and 8 (RX) for GSM communication.

---

5. TinyGPS++.h

- **Function:**
  A lightweight GPS parsing library.
- **Purpose:**
  Converts raw NMEA GPS data into readable values like:
  - Latitude
  - Longitude
  - Altitude

- Number of satellites
- Speed, etc.

- **Used for:**
  Getting real-time location data from the GPS module.

5. System Features and Operation

The system works through a series of steps [1-3,8] that can be explained through the following block diagram:

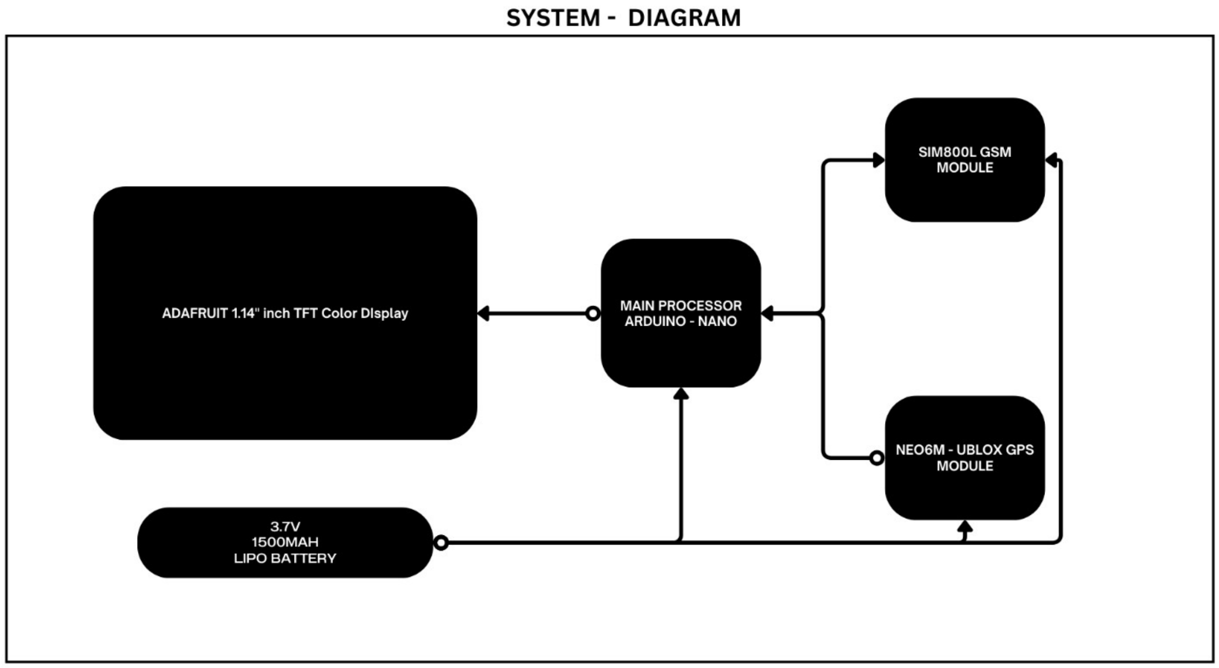

SYSTEM - DIAGRAM
SIM800L GSM MODULE
ADAFRUIT 1.14" inch TFT Color Display
MAIN PROCESSOR ARDUINO - NANO
NEO6M - UBLOX GPS MODULE
3.7V 1500MAH LIPO BATTERY

Figure (7): The block diagram shows the control flow from the Arduino – controlling the SIM800L Module and the UBLOX GPS MODULE, and displaying essential data on the Adafruit TFT Display, the whole system is powered through a 3.7v – 1500mAh LiPo Battery

- **NEO6M U-BLOX GPS Module**
  Retrieves GPS location – (Latitude, Longitude, Altitude) and Satellites locked onto.
- **SIM800L GSM Module**
  Uses UART interface to communicate with the Arduino – Receives SMS AT commands – "GETLOC" for location fetching and sends the map link to the corresponding phone number, this is based on GPRS Communication.

- **Adafruit 1.14" inch TFT Display**

Displays vital **data** for the user :

-Positional Longitude.

-Positional Latitude.

-Positional Altitude.

-No. of Satellites locked onto.

5.1. System Performance Metrics

Table 1.

COMPONENT	**STARTUP (s)**	**Voltage Rating(V)**	**Power Consumption (W)**
Arduino Nano V3	*~1s*	*~9V – 12V*	*~0.52W – 0.845W*
GPS Module	*~60s-120s*	*~3.7V – 5.5V*	*~0.165W – 0.33W*
GSM Module	*~10s*	*~3.2V – 4.0V*	*~0.0185W– idle mode* *~2.29W – Peak*

The following table (Table 1) demonstrates the startup time [1-3], voltage operation ratings and power consumption of each of the main components of the tracker – showing the maximum power consumption of the system to be = ~ 4.0W.

6. Conclusion

In this report, we have presented the design and implementation of a low-power GPS tracking system that uses GSM communication for location sharing. By integrating the U-Blox GPS module for accurate geolocation data gathering and the SIM800L GSM module for data transmission, the system is able to provide reliable tracking capabilities with minimal power consumption. The SMS-based communication approach reduces infrastructure complexity and cost while delivering on-demand location updates, making it a practical solution for various tracking applications, where live tracking isn't necessary.

The experimental results have shown that the system performs moderately well under different dynamic conditions – optimally in the outdoors, with reliable communication through GSM and stable power consumption.

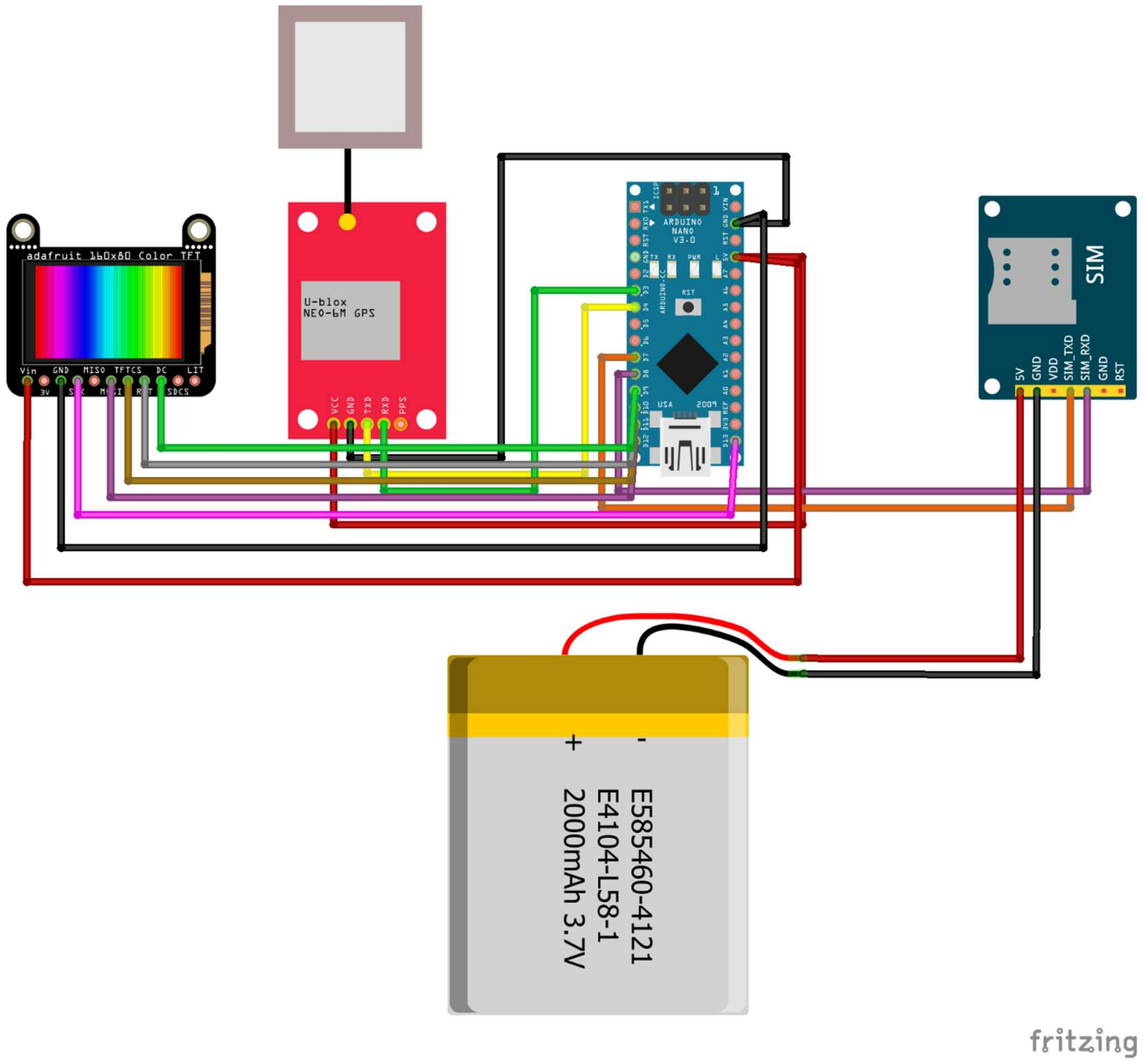
Overall, the proposed low-power GPS tracker proves that high-performance location-sharing can be achieved with a focus on low-cost infrastructure, and low power consumption. Future improvements may focus on optimizing power management further and exploring advanced features, such as real-time tracking through mobile applications or backend servers for enhanced tracking capabilities.

Acknowledgements

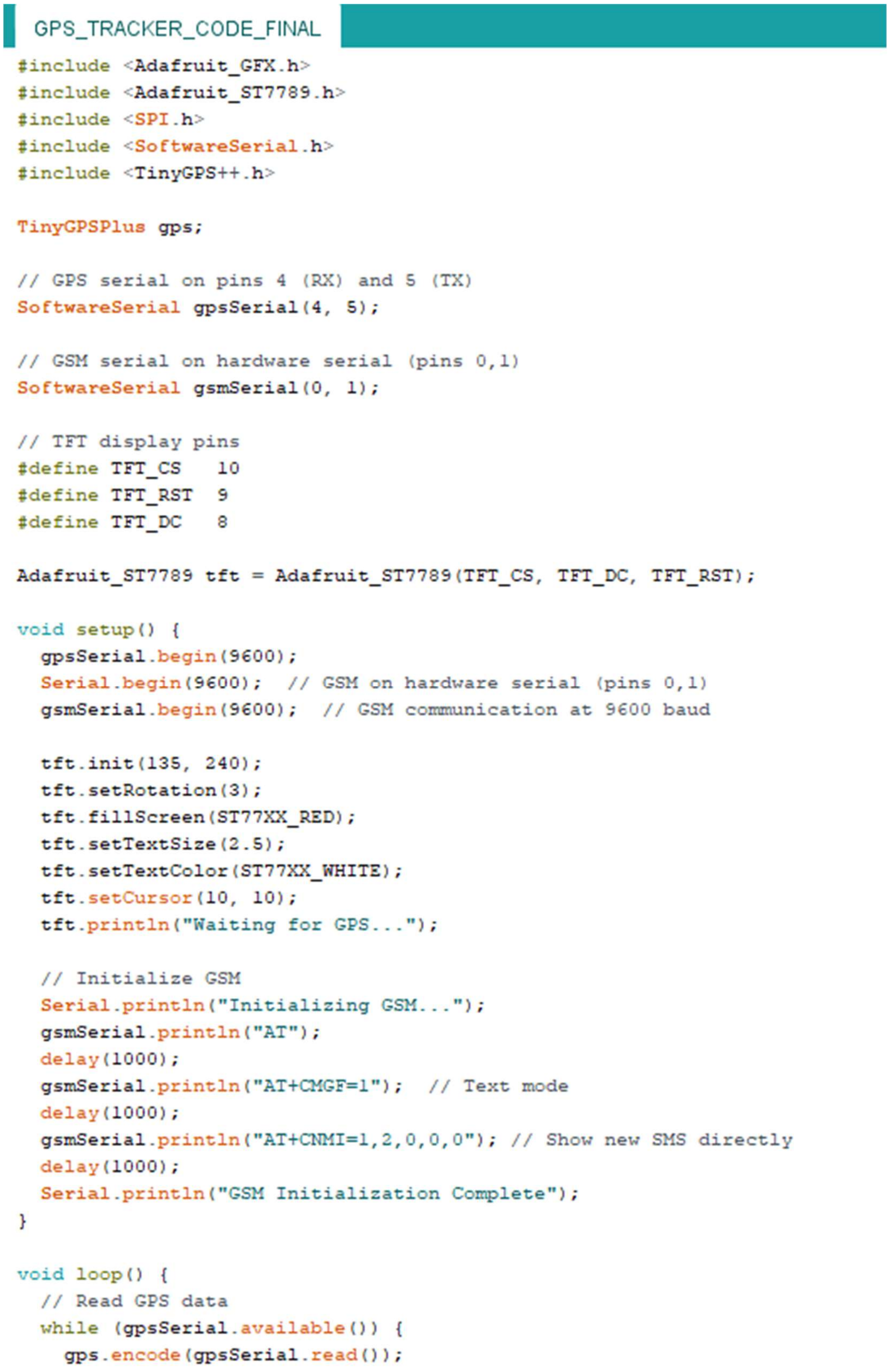
This work was supported by Dr. Hatem Diab, Head of the Electrical Energy Engineering Department, AASTMT.

APPENDICES

APPENDIX (A): The Schematic Diagram of the GPS Tracker Device

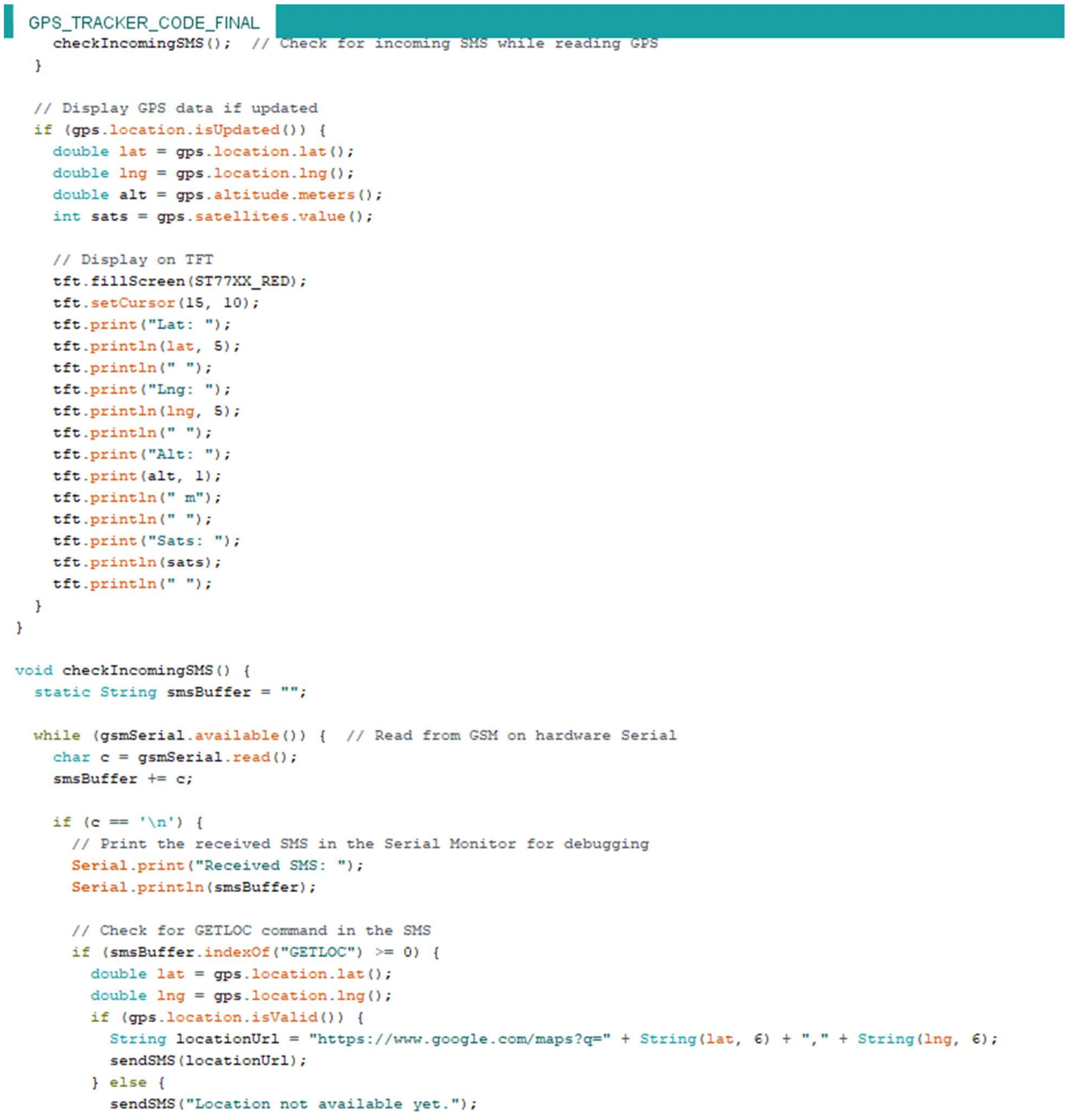


APPENDIX (B): The Arduino Code for GPS Tracker

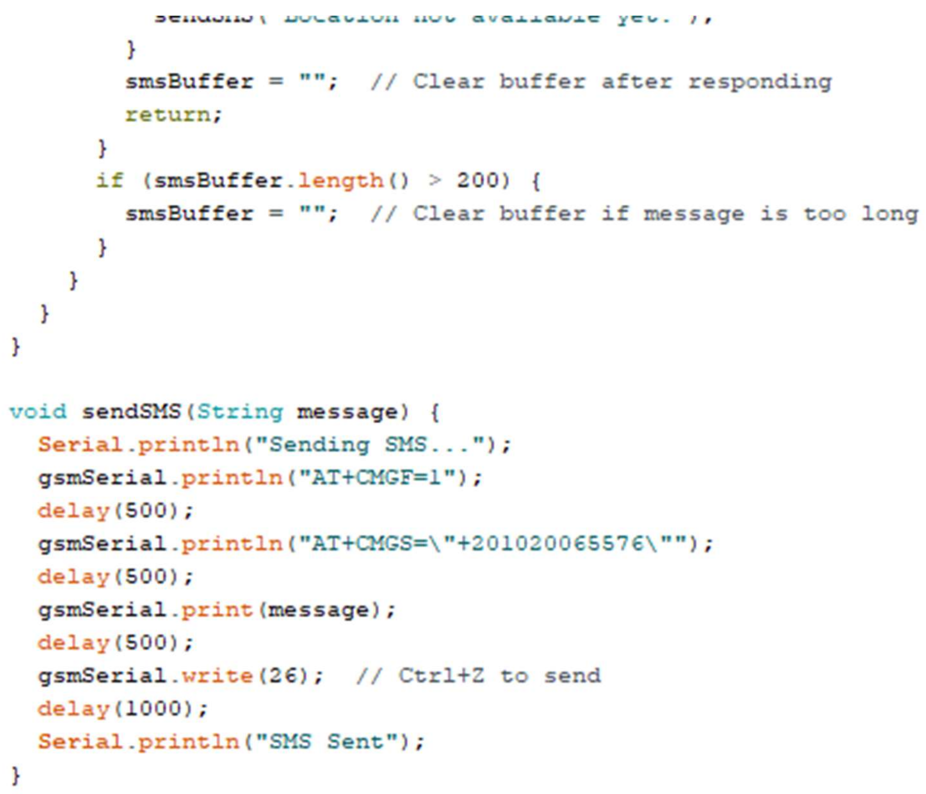


GPS_TRACKER_CODE_FINAL

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7789.h>
#include <SPI.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

TinyGPSPlus gps;

// GPS serial on pins 4 (RX) and 5 (TX)
SoftwareSerial gpsSerial(4, 5);

// GSM serial on hardware serial (pins 0,1)
SoftwareSerial gsmSerial(0, 1);

// TFT display pins
#define TFT_CS   10
#define TFT_RST  9
#define TFT_DC   8

Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);

void setup() {
  gpsSerial.begin(9600);
  Serial.begin(9600);  // GSM on hardware serial (pins 0,1)
  gsmSerial.begin(9600);  // GSM communication at 9600 baud

  tft.init(135, 240);
  tft.setRotation(3);
  tft.fillScreen(ST77XX_RED);
  tft.setTextSize(2.5);
  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(10, 10);
  tft.println("Waiting for GPS...");

  // Initialize GSM
  Serial.println("Initializing GSM...");
  gsmSerial.println("AT");
  delay(1000);
  gsmSerial.println("AT+CMGF=1");  // Text mode
  delay(1000);
  gsmSerial.println("AT+CNMI=1,2,0,0,0"); // Show new SMS directly
  delay(1000);
  Serial.println("GSM Initialization Complete");
}

void loop() {
  // Read GPS data
  while (gpsSerial.available()) {
    gps.encode(gpsSerial.read());
```



```
    checkIncomingSMS();  // Check for incoming SMS while reading GPS
  }

  // Display GPS data if updated
  if (gps.location.isUpdated()) {
    double lat = gps.location.lat();
    double lng = gps.location.lng();
    double alt = gps.altitude.meters();
    int sats = gps.satellites.value();

    // Display on TFT
    tft.fillScreen(ST77XX_RED);
    tft.setCursor(15, 10);
    tft.print("Lat: ");
    tft.println(lat, 5);
    tft.println(" ");
    tft.print("Lng: ");
    tft.println(lng, 5);
    tft.println(" ");
    tft.print("Alt: ");
    tft.print(alt, 1);
    tft.println(" m");
    tft.println(" ");
    tft.print("Sats: ");
    tft.println(sats);
    tft.println(" ");
  }
}

void checkIncomingSMS() {
  static String smsBuffer = "";

  while (gsmSerial.available()) {  // Read from GSM on hardware Serial
    char c = gsmSerial.read();
    smsBuffer += c;

    if (c == '\n') {
      // Print the received SMS in the Serial Monitor for debugging
      Serial.print("Received SMS: ");
      Serial.println(smsBuffer);

      // Check for GETLOC command in the SMS
      if (smsBuffer.indexOf("GETLOC") >= 0) {
        double lat = gps.location.lat();
        double lng = gps.location.lng();
        if (gps.location.isValid()) {
          String locationUrl = "https://www.google.com/maps?q=" + String(lat, 6) + "," + String(lng, 6);
          sendSMS(locationUrl);
        } else {
          sendSMS("Location not available yet.");
```



```
          sendSMS("Location not available yet.");
        }
        smsBuffer = "";  // Clear buffer after responding
        return;
      }
      if (smsBuffer.length() > 200) {
        smsBuffer = "";  // Clear buffer if message is too long
      }
    }
  }
}

void sendSMS(String message) {
  Serial.println("Sending SMS...");
  gsmSerial.println("AT+CMGF=1");
  delay(500);
  gsmSerial.println("AT+CMGS=\"+201020065576\"");
  delay(500);
  gsmSerial.print(message);
  delay(500);
  gsmSerial.write(26);  // Ctrl+Z to send
  delay(1000);
  Serial.println("SMS Sent");
}
```

REFERENCES

[1] U-Blox AG, *NEO-6 GPS Modules Data Sheet*, U-Blox AG, Switzerland, 2012. [Online]. Available: https://www.u-blox.com/en/product/neo-6-series

[2] SIMCom Wireless Solutions Ltd., *SIM800 Series Hardware Design*, SIMCom, 2015. [Online]. Available: https://simcom.ee/documents/SIM800L/SIM800L_Hardware_Design_V1.00.pdf

[3] Arduino, *Arduino Nano Documentation*, [Online]. Available: https://docs.arduino.cc/hardware/nano

[4] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*, 2nd ed., Wiley-Interscience, 2007. doi:10.1002/0470099720

[5] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications*, 2nd ed., Artech House, 2005. ISBN: 978-1580538947

[6] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed., Pearson, 2011. ISBN: 978-0132126953

[7] C. Elliott and R. Wright, "Understanding and Applying AT Commands," *National Instruments*, 2004. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/06/understanding-and-applying-at-commands.html

[8] Adafruit Industries, *1.14" Color TFT Display (ST7789) Product Guide*, [Online]. Available: https://learn.adafruit.com/adafruit-mini-tft-breakout

[9] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, Ganga-Jamuna Press, 2006. ISBN: 9780970954428

[10] D. P. Agrawal and Q.-A. Zeng, *Introduction to Wireless and Mobile Systems*, 3rd ed., Cengage Learning, 2010. ISBN: 978-1439062068