BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

GPS TRACKER WITH ONLINE DATA VISUALISATION
GPS TRACKER S ONLINE VIZUALIZACÍ DAT

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR   Mahmoud Itmaizi
AUTOR PRÁCE

SUPERVISOR   Ing. Jan Najman
VEDOUCÍ PRÁCE

BRNO 2020

Specification Bachelor's Thesis

Department:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Mahmoud Itmaizi**
Study programme:	Applied Sciences in Engineering
Study branch:	Mechatronics
Supervisor:	**Ing. Jan Najman**
Academic year:	2019/20

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Bachelor's Thesis:

GPS tracker with online data visualisation

Concise characteristic of the task:

The aim of this work is to create a device that can be used to track GPS position with sending data to the server. The motivation for the development of this device is that commercially available solutions are usually difficult to modify, which complicates the use of customised data processing solution or change of the module settings. It is planned to use the already available modules to build the device. In addition to storing and displaying data, the web application should be able to easily extend functionality in the future. Ideally, the device should be as small as possible and powered from the battery.

Goals Bachelor's Thesis:

1. Review available hardware and software solutions.
– Overall dimensions as small as possible.
– Low power consumption and independent power supply (battery).
– Programmable control unit.
– Web application / server for receiving, storing and displaying measured data on the web.
2. Select and assemble components and program sending position data.
– Send data via serial line.
– Send data to the web application, including map display (should work in any browser).
3. Test the functionality and design a suitable box.
– Signal stability in various conditions (urban area, forest, etc.).
– Influence of the type and location of antennas on the function of the device.
– Protection of components in the box against mechanical damage and shocks.
4. Extend the solution with additional functionality.
– Basic controls and indicators (button, LED).
– Save and display motion history (track) within the specified range.
– Support for multiple devices and simultaneous display of data on the web.
– Sending additional information – battery status, signal strength, movement speed.
– Change module settings – update frequency, standby, etc.

Recommended bibliography:

WELLING, L., THOMSONOVÁ, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.

HOJGR R., STANKOVIČ J.: GPS Praktická uživatelská příručka. Computer Press, 2007. ISBN: 978-80-251-1734-7.

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2019/20

In Brno,

L. S.

....

prof. Ing. Jindřich Petruška, CSc.   doc. Ing. Jaroslav Katolický, Ph.D.
Director of the Institute   FME dean

ABSTRACT

The aim of this work is to design a GPS device capable of getting the GPS data and sending it over the GPRS mobile network to a web application, to be stored, and displayed on a map. the designed device allows for modification and changing of setting. this project reviews the hardware and software for the device and the web application.

Abstrakt

Cílem této práce je navrhnout zařízení GPS schopné získávat GPS data a odesílat je

přes mobilní síť použitím GPRS do webové aplikace, která se má uložit a zobrazit na mapě.

Navržené zařízení umožňuje úpravu a změnu nastavení. tento projekt přezkoumává

hardware a software pro zařízení a webovou aplikci

Rozšířený Abstrakt

Navigační systémy jsou již dlouho implementovány, aby vedly lidi na jejich cestách. Počatek bylo průvodcem po hvězdách a používáním kompasu až používání systému GPS který známe dnes. Tato revoluce vylepšilo lidské schopnosti cestovat a provozovat podniky i zachraňit životy.

Satelitní navigační systémy dnes nabízejí globální pokrytí a jsou otevřené pro kdokoli který má GPS přijímač. Tyto GPS přijímače mohou být v různých tvarech a velikostech na základě jejich funkce.

V tomto projektu se podíváme na satelitní navigační systémy a popíšeme pracovní principy a jaký je rozdíl mezi systémy GPS a GNSS. Poté se podíváme na hardwarové komponenty potřebné k výrobě GPS sledovač zařízení, tyto komponenty zahrnují řídící jednotku a GPS modul. Pro zasílání těchto dat přes internet, kde mají být uložené a zobrazané, je zapotřebí také modul GSM. Pak Webova stranka je vytvořena takže přijatá GPS polohy přes internet uloží je do databáze a zobrazuje je na mapě.

Aby bylo možné dosáhnout cílů projektu, byly komponenty vybrány jako řídicí jednotka a modul GPS/GPRS. Zvolena řídicí jednotka je Arduino Nano, která má má malou velikost a má nízkou spotřebu energie. Module GSM/GPRS Sim808 byl použití, díky svým schopnostem GPS a GPRS, tj. Schopnosti přijímat polohu GPS a odesílat informace přes mobilní síť.

Na kontrolu Sim808 Arduino posílá AT příkazy do Sim808, aby určil požadovanou operaci, zatímco Arduino přijímá data o poloze z GPS Sim808 funkce. Poté Arduino kód analyzuje data a vybere požadované hodnoty, aby je znovu poslal do Sim808 k odeslání přes GPRS službu na internetu

Zařízení je vyrobeno tak, aby bylo napájeno baterií. Baterie použitá pro toto zařízení byla 3800 mAh, která je schopna dlouhodobé funkčnosti zařízení. Zařízení má více elektrických komponent (LED, tlačítka, kondenzátory… atd.). Všechny komponenty jsou vloženy do malé krabičky vytištěné technologií 3D tisku.

Webová stránka byla vytvořena pomocí infuzní Google map, která umožňuje zobrazit polohy na mapě, s databází k uložení dat. Webová stránka byla vytvořena pomocí tří webových programovacích jazyků, PHP, JavaScript a HTML. každý z nich pro určitý účel, ale hlavní jazyk pro označení polohy je JavaScript kvůli použití "Google JavaScript map API" což je nástroj pro interaktivní mapy.

Testování zařízení ukázalo stabilní a pravidelné přijímání a odesílání GPS polohy bez problémů. Testování finální fáze webové stránky ukázalo stabilní uložení dat a zobrazení na mapě.

Testování přesnosti polohy bylo provedeno pohledem na výsledného zobrazení na mapě a jeho porovnáním se skutečnou trasou, přesnost byla dostatečně přesná pro mnoho aplikací a v otevřených prostorech může mít umístění odchylku maximálně tři metry. Přesnost poloha závisí na počtu pozorovaných GPS satelitů a poloze antény.

KEYWORDS

Klíčová Slova

Bibliography

ITMAIZI, Mahmoud. *GPS tracker s online vizualizací dat.* Brno, 2020. Dostupné také z: **https://www.vutbr.cz/studenti/zav-prace/detail/121527**. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jan Najman.

Declaration of authenticity

I declare that I have elaborated my bachelor's thesis on " GPS tracker with online data visualisation" independently with the use of technical literature and other sources of information which are all quoted in the thesis and detailed in the list of references at the end of this thesis.

In Brno on 26.6.2019   Mahmoud Itmaizi

Acknowledgement

TABLE OF CONTENTS

1.   Introduction

GPS tracking devices are very important tools that we use in our everyday life. Thanks to the technological advancement we have access to the amazing positioning system that enhance our lives for the better. GPS trackers can provide extra measure of security for children, pets, and property such as vehicles by providing their exact location, and in many cases, it can save lives.

Although there are many types of GPS tracking devices, they are rarely modifiable. Usually GPS trackers do not allow for customization or enabling extra features that might be wanted.

This work is guided for making a GPS tracking device capable of sending GPS data over the mobile network, and making a Web application which can be customizable. This paper is divided into two main parts

The research part which describes the navigation systems and their history and their working principles. Then it describes the hardware components and the reviewed hardware options. After that it reviews the possible web applications suited for this project.

The practical part describes the processes taken towards a functional system of GPS tracking device, as well as the web application. Steps start from type and design of the hardware components, which include control unit, battery, GPS module and suitable box to save them from damage. Then to the software part of coding and controlling the hardware. Later it shows the Web application and how it was implemented. And once the device implementing is cleared comes the testing and results to assess the device functionality and reliability.

2. Research

This chapter reviews the theory part of the paper, and it describes the functions of each part in the whole project, including the positioning systems GPS and GNSS and the available hardware and software that can be used for such project and how they all work together. As well as some of the review for Web application possibilities.

2.1. Satellite Navigation Systems

Satellite navigation systems have a huge impact on our today lives. This part describes the history and functionality of the GPS system and the GNSS. The satellite navigation systems are satellite systems that send radio signals which provides data for receiving devices to calculate their location (longitude, latitude, and altitude) based on location of satellites and time differences.

2.1.1. GPS and GNSS

GPS

GPS is the abbreviation for (Global Positioning System), which is a satellite-based navigation system made by the United states. Originally limited to use by the United States military, the GPS project started in 1973, the plan included 24 satellites to orbit the earth. The 24 satellites arrangement is to ensure there are 4 satellites in the view virtually from any point on the planet [2]. The first satellite launched in 1978 and the full constellation of 24 by 1993. The GPS was open for civilian use in 1983 when U.S. President Ronald Reagan announced that GPS would be made available for civilian uses once it was completed.[1]

GPS satellites orbit the earth at an altitude of 20200 KM in the medium earth orbit, while each satellite circles the earth twice a day. The 24 satellites arrangement is to ensure there are 4 satellites in the view virtually from any point on the planet [2]

 In 2011 3 more satellites were added to the system, to operating with 27 satellites to improve coverage. As of February 20, 2020, there were a total of 31 operational satellites in the GPS constellation, not including the decommissioned (on-orbit spares).[2]

GNSS

The GNSS is the abbreviation for (Global Navigation Satellite System), and it represents all the navigation systems with global coverage opened for use. These navigation Systems today are:

GPS(USA): As described before is the first navigation system, and has a global coverage, was and still operated by the US military, yet it is open for civilian use. The GPS can provide accuracy about 3-5 metres in average and down to 30 cm. [3]

GLONASS(Russia): The Russian global navigation system, the abbreviation is taken from "GLObal NAvigation Satellite System". Operated by Russian Aerospace Defence Forces. [4] Started development on 1976, then completed on 1995 but due to loss of satellites more satellites needed to be launched. on 2011 the full global coverage was achieved. These satellites orbit at 19100 km altitude. The GLONASS is suited for high latitudes (north). This satellite system is also made for army use at beginning but later it was made for civilian use as well.[5] GLONASS supports horizontal positioning accuracy within 5–10 meters and 15 metres in the vertical positioning. [6]

Galileo (EU): Is the global navigation system created by the European union through the European GNSS agency (GSA) headquartered in Prague-Czech Republic.[7] Started operating in December 2016. First satellite launched in 2005, but first operational satellite launched in 2011. And In 2018 the Galileo constellation increased up to 26 satellites extending the global coverage. [8] This system reached horizontal positioning accuracy of 4 metres, and 8 metres for vertical positioning.[9]

BeiDou(China): The Chinese global navigation system (BDS). Started in year 2000 with experimental satellites that offered limited coverage only for china and some surrounding areas and decommissioned at the end of 2012. Then a new satellite constellation started operation from 2011 with 10 satellites. Satellite launching continued and will eventually consist of 35 satellites and is expected to provide global services upon completion in the end of 2020. With expected accuracy of 5-10 metres. [10][11]

Additionally, there are two regional systems – QZSS (Japan) and IRNSS or NavIC (India). These systems help GNSS the GNSS Systems on those regions.

2.1.2.Working Principle of GPS:

The main concept of satellite positioning relies on the position of satellites and timing. Each of these satellites use an atomic clock for a high precision time measuring, and all of these clocks are synchronised with each other. Each satellite must have its exact location on its orbit (known as the Ephemeris). Meanwhile date, time and location of the satellite are being sent continuously from the satellites as well as the information about the position of orbits of all satellites (known as the Almanac).

Each GPS satellite continuously transmits a radio signal containing the current time and data about its position. Since the speed of radio wave is constant (speed of light) the time delay between when signal was sent and when it was received can tell us the distance from the satellite to the receiver.

To calculate the position on earth the GPS receiver needs to know each satellite position, and the distance from each satellite and the time. To achieve this, 4 satellites are needed to be in view. Then using 3-D Trilateration equation which basically calculate the intersection of the 4 spheres (each sphere represents the distance from each satellite), this intersection is the position of the GPS receiver as can be seen in the figure 2.1. The position is determined by (longitude, latitude, altitude)[12]

But for the distance to be calculated the clock on the GPS device must be synchronized with the clocks of the satellites for time accuracy, for example if there was a time inaccuracy of 38 microseconds(the daily effect of relativity theory) would cause inaccuracy of 10km. So, the timing of the GPS receiver must be synchronized with the satellites to high level of accuracy.

When there are 4 satellites in the view, the GPS receiver makes trilateration and when all 4 spheres do not intersect in one point(due to unsynchronized timing), the receiver can easily calculate the necessary time adjustment that will cause the four spheres to intersect at one point. This time is the exact synchronized time of the satellites. The receiver does this constantly whenever it is on, which means it is nearly as accurate as the atomic clocks.[12]

If the GPS receiver is only able to get signals from 3 satellites, it can still get the position, but it will be less accurate. As the GPS receiver needs 4 satellites to work out the position in 3-dimensions. If only 3 satellites are available, the GPS receiver can get an approximate position by assuming that the GPS receiver is at mean sea level.[13]
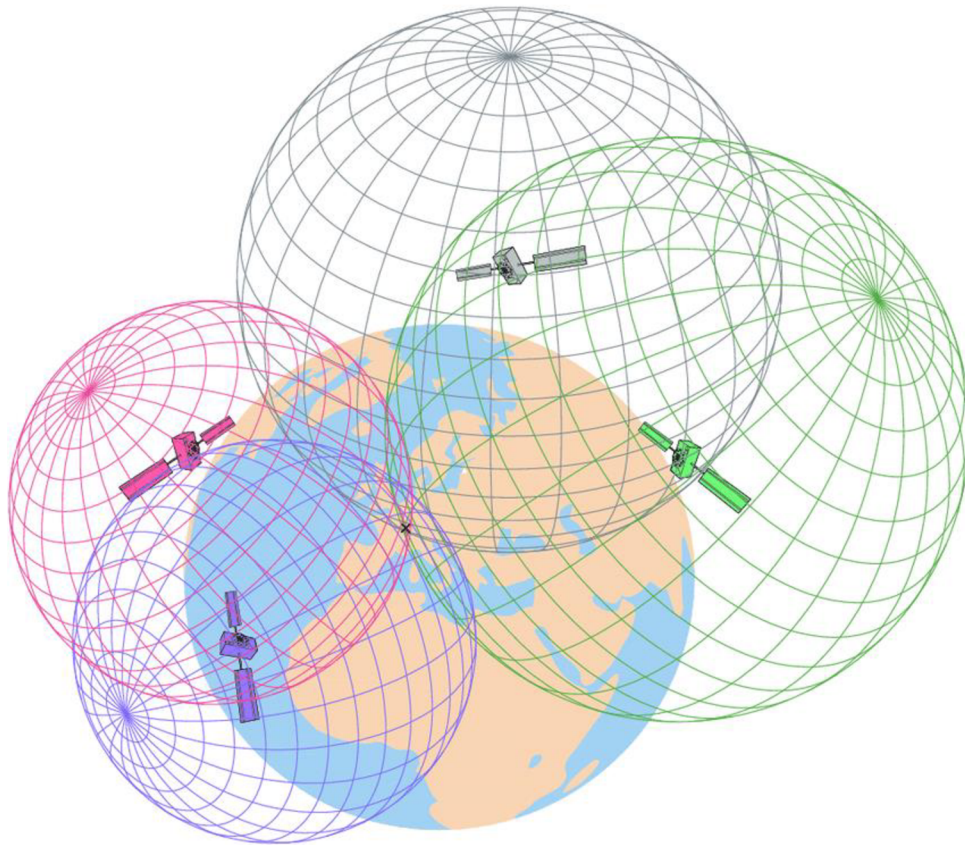
Figure 2.1 Position computation by trilateration [32]

It is also important to note that there are several techniques being taken to improve GPS accuracy. Like some improvement to the GPS receivers making them able to download the Almanac data(the orbital positions of all satellites) from the cellular network so the GPS receiver can have a hot start and within few seconds it can figure the position, which can take up to15 minutes without this technique.

Another improvement is the **DGPS (Differential Global Positioning System)**, which is an enhancement of the position accuracy and to apply fixes. DGPS is a network of a fixed ground-based stations that broadcast the difference in position calculated from the satellites and the known fixed position of the station. the difference is being broadcasted from these stations to be received by GPS receivers to apply fixes for better accuracy. The figure 2.2 shows how it works.
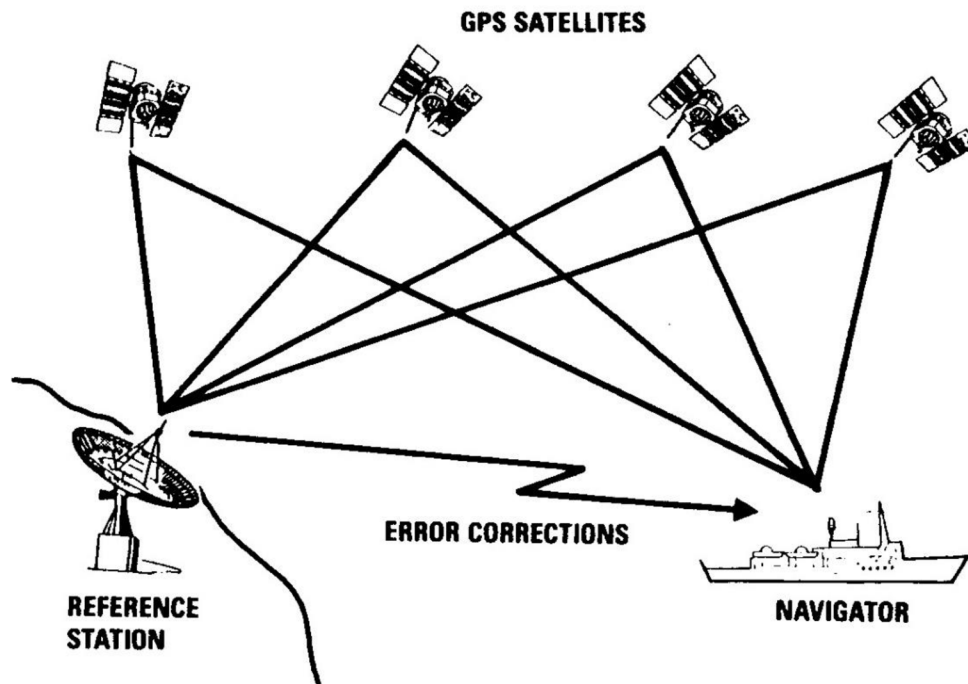
figure 2.2 Differential Global Positioning System [33]

2.2. Microcontrollers and GPS/GSM modules:

For the sake of this project it was needed to have a hardware device capable of receiving GPS data, and picking the wanted data then sending it over the mobile network. This device can be made of a GPS receiver, control unit, and GSM mobile network module for sending the data to the server.

2.2.1.Microcontroller:

This is the control unit that will control the processes inside the final device. It allows for sending commands to other modules, and picking the wanted data and shaping it. The microcontroller also can be programmed to do these functions as desired, leaving a lot of space for modification for the final GPS device.

There are a lot of microcontrollers available on the market each with its special features that can suit the desired project. For this project it is evident that the desired features are:

- Low power consumption: due to the plan of making the finished device powered by a battery.

- Good enough memory: The memory needed will be few kilobytes and its role is to save the code in the machine language, so no need for a huge memory.

- Low price: the lower the price is the better. This can help if the project would be expanded to multiple devices. As well as if some mistake happens during development it will not cost a lot to replace the microcontroller.

- Small size: if the desired features exist in a smaller microcontroller, there is no need to use a bigger one since the final device should be as small as possible for extra usability.

- Durability: the microcontroller should be durable to extend the overall life of the final device, and to withstand some minor mistakes done during development.

By looking at these desired features, Arduino board was chosen for this project sake. The most common microcontroller for Arduino boards is the Atmega328, which is used on the Arduino UNO, Nano which are the most common. Here we can review common Arduino boards features:

- Availability: Arduino boards can be easily found on most stores related to DIY and electronics, and online shops.

- Availability of open source code: the Arduino community is quite a big one, this comes in handy for trials and development of the project.

- On-board regulator: reducing the need for an external voltage regulator and giving extra safety measure.

- Simple to use: amongst all microcontrollers Arduino is one of the simplest to use, and it uses the Arduino programming language which is a set of C/C++ functions [14]

- Low power consumption: consumes 19mA at 5v based on the official website for the Arduino nano, but power consumption varies depending on the status and the type of Arduino.[15]

- Many input/output pins: Arduino boards offers multiple pins for digital(input/output) and Analog(input) that give the opportunity for adding more electronic components (sensors, LEDs, ….) and connecting to multiple modules.

Adding to this, it is important to note that there are other options like:

Raspberry pie (have a huge power consumption compared to Arduino)

PIC microcontroller (Arduino is superior to the PIC for a 1device but in case of scaling PIC will be more reliable, and there a lot more online forums for Arduino to learn from and to solve issues that might occur), but the Arduino seems to be sufficient enough for this project.

2.2.2.GPS module:

The word module usually relates to the meaning of "part", and here we mean by it a chip or a circuit that is a part of a bigger device. The GPS module is the hardware component that is responsible for receiving the GPS data and processing it to give the GPS position. It is mainly composed of 2 parts:

Antenna: the part that is responsible for receiving the GPS signals from the satellites, the position of antenna is the most important for positioning accuracy.[16] antennas can be passive or active (with an active amplifying element)

Processing unit: the part where the distances, time, and position are being calculated.

Each GPS module (also known as GPS receiver) is made based on a navigation system or a combination of them. meanwhile most GPS receivers today are using the GNSS system which uses all the 4 navigation systems at once. But the term GPS is still the most famous so even GNSS modules are being called GPS modules.[17]

Reviewed GPS module:

Neo7m GPS module

This module can determine the position based on GPS/GLONASS satellites system.[21] Neo7m GPS module is easy to use and straight forward, it would send the position and time data at high rates. The figure 2.3 shows the Neo7m GPS module

figure 2.3 Neo7m GPS module [34]

2.2.3. GSM/GPRS module:

This is the hardware component responsible for connecting, sending and receiving of data using the mobile network (GSM and GPRS systems). To communicate with the GSM/GPRS modules AT commands are used. the module usually contains 3 parts:

Antenna: for receiving and sending of data signals.

Processing unit: responsible for establishing the communication with the mobile network.

Sim card: stands for (Subscriber Identity Module), and it sets an identity for the user to communicate on the mobile network with it.[18]

GSM

GSM stands for (Global System for Mobile Communications), is a standard developed by the European Telecommunications Standards Institute (ETSI). It was created to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones and is now the default global standard for mobile communications.[18]

GPRS

GPRS stands for (General Packet Radio Service) is a standard protocol compatible with 2G, 3G networks. It extends the GSM capabilities and enables always-on internet access as well as Multimedia Messages and other advanced features and services.[19]

AT commands

The AT commands are used with GSM and GPRS modems and modules or phones to access services and information. AT stands for attention. For example, to ask the GSM module to send a message the AT command (AT+CMGS) is sent to the module.[20]

Reviewed module:

Sim900 GSM/GPRS module

The reviewed variety of this module was the sim900 GSM/GPRS shield module, which is made to be a shield module for Arduino UNO. This module allows for internet connection using the GPRS. Sim900 allows for sending data over the internet, sending messages, and making voice calls. the Sim808 shield is shown in the figure 2.4
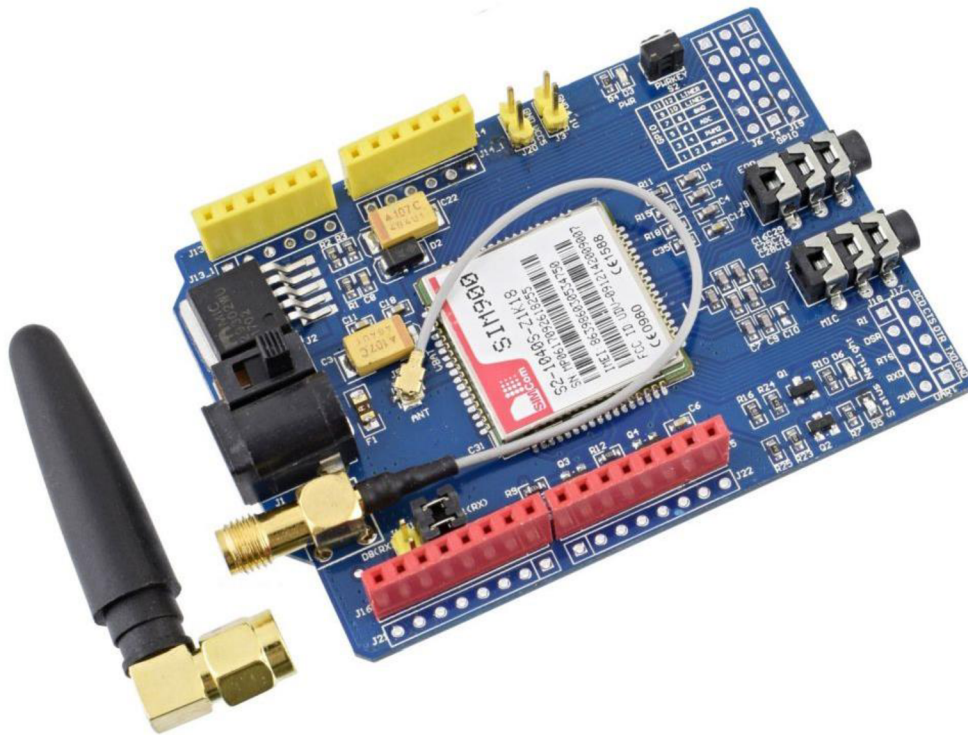
Figure 2.4 Sim900 Shield module [35]

2.2.4. GPS/GPRS module

Another possibility is to use two-in-one GPS and GPRS module. This type of modules has the functions of a GPS module and a GSM/GPRS module in the same time. the reviewed module was the Sim808 GSM/GPRS and GPS module.

Sim808 GSM/GPRS and GPS module

This module has both capabilities of the previous modules, it can receive and figure out the GPS location, as well as sending/receiving data over the mobile network. it is important to mention that Sim808 can offer a big reduce on size of the final device by replacing two modules with one. Both GPS and GPRS functions are controlled by AT commands. The Sim808 is shown in figure 2.5
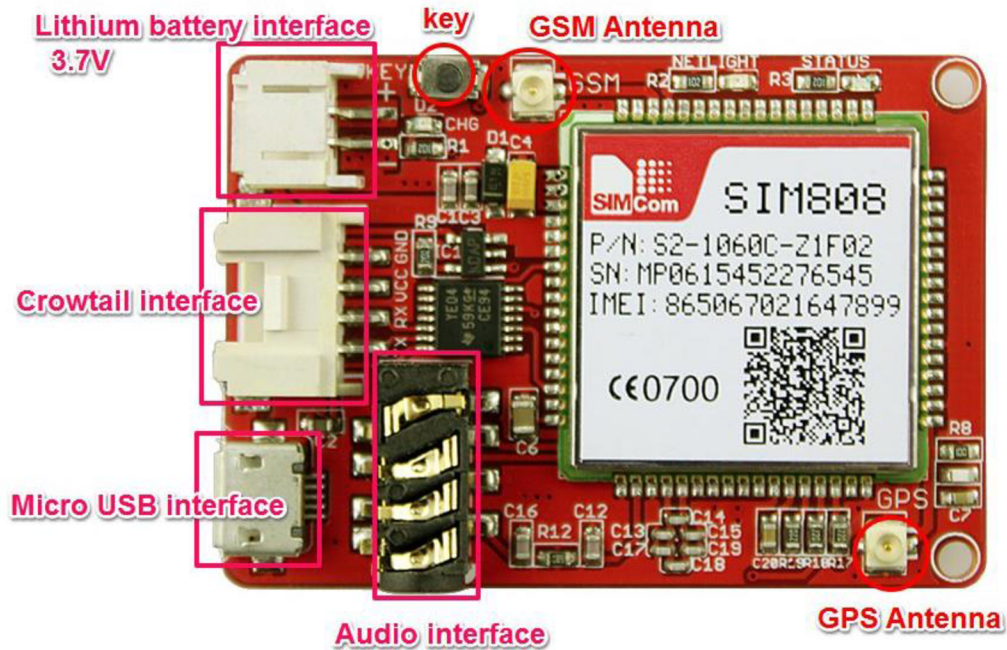
figure 2.5 Sim808 model number : CT0078SIM [36]

Here are some of the general features of the Sim808:

- Low power consumption

- Quad-band 850/900/1800/1900MHz

- Horizontal position accuracy: less than 2.5m

- Time To First Fix:

  - Cold starts: 30s (typically)

  - Hot starts: 1s (typically)

  - Warm starts: 28s (typically)

- AT cellular command interface

- Supports charging function.[22]

Power consumption of the Sim808

Inside the Sim808 chip there are multiple engines, we will look at the GPS engine and the GSM engine current consumption at supply voltage 3.4 ~ 4.4V.

The GPS engine consumes: 24 mA on the continuous tracking mode [23] page 13.

The GSM engine consumes:

- 16.8 mA on the idle mode (AT+CFUN=1) which is set for this project.

- 140 – 220 mA for voice call.

- 160 – 445 mA for Data (internet connection) depending on the modes. [23] page 63.

2.3. Available Web Applications for Reviewing the GPS data.

Once the GPS data have been sent on the internet, there should be a mean of viewing these data and maybe saving it. The GPS data can be viewed as a table of latitudes, longitudes, date…. so on all viewed into a table, but that is not efficient. So, the best representation of GPS data will be on a map.

Here are some of the application that consider a map for receiving, viewing, and saving of the GPS data:

2.3.1. IoT Platforms

IoT stands for (Internet of Things), which is a network of devices connected via internet and sharing data with a server.

An IoT platform is one system of connected devices, where their data and information can be viewed from anywhere as well as sending commands to them. Each set of sent data from these devices can be viewed as the user wants (Graph, pie chart, …. etc). A typical configuration for the sake of this project is when the final device sends GPS data to an IoT platform that has a map, then the map with positions will be marked and viewed from anywhere.

2.3.2. Websites:

Building a website with built-in map can also be another option for storing and viewing of the GPS data. This works by infusing a map from a maps provider like google maps into a website, then asking the map provider to add markers, lines, information window and many other options to the map based on the saved information that being sent form the GPS device.

Storing

Once the GPS data are received by the website, it should be saved somewhere. The term for the place where the data being saved is Database. There are many types of data bases depending on he needs and accessibility. Among databases are :

- **MySQL Database**: The data in a MySQL database are stored in tables. Were table is a collection of related data, and it consists of columns and rows. MySQL is the most popular database system used with PHP.

- **Flat-file database** : is a database stored in a file called a flat file. Records follow a uniform format, and there are no structures for indexing or recognizing relationships between records. The file is simple and can be a plain text file. Usually the saved valued are saved in lines with separating of each value using commas, tabs…. etc.

Viewing

Once the data are saved, they are passed to a the map within the website, and each position will be marked as desired.

There are many map providers like:

- Google maps: well known, offers a lot of documentations for how to infuse a google map into a website, offers a lot of types of services related to maps and with so many developing options. And has a big community for issues solving.

- Leaflet: its defined as (an open-source JavaScript library for mobile-friendly interactive maps), gets a lot of recommendation from developers.

- Mapy.cz: a Czech version of maps providers, that offers the possibility for infused maps into webpages.

It is important to note that there are so many maps providers, but these are the ones reviewed as potential possibilities for this project.

Map tool:

The map providers mentioned above, have many map tools that can be infused into the website, the most usable for the GPS map applications is the "JavaScript map API", which Adds an interactive map to the website, and allows for customizations.

Other tools can be more directed to use of mobile phones, or can be static maps without the ability to interact, like the "Static map API"

3. The practical part:

In this Part we will take steps towards a fully working project starting from the hardware components to the storing and viewing of location data on a web application.

3.1. Hardware design:

3.1.1.Chosen Microcontroller:

In the research part it was cleared why to choose Arduino for this project, but then it was the task of finding which Arduino board to be chosen. This project started with Arduino UNO, which is the most famous among all Arduino boards, but the problem was it is big and consume more power than other Arduino boards.

After reviewing some of the boards, Arduino Nano was chosen shown in figure 3.1, and it is for these reasons:

- Small sized: it is the smallest of all Arduino boards with 18 x 45 mm (about the quarter of the size of Arduino uno).

- Enough input/output pins: 8 analog pins, and 22 digital pins, giving the possibility for connecting Sensors, LEDs, modules…etc.

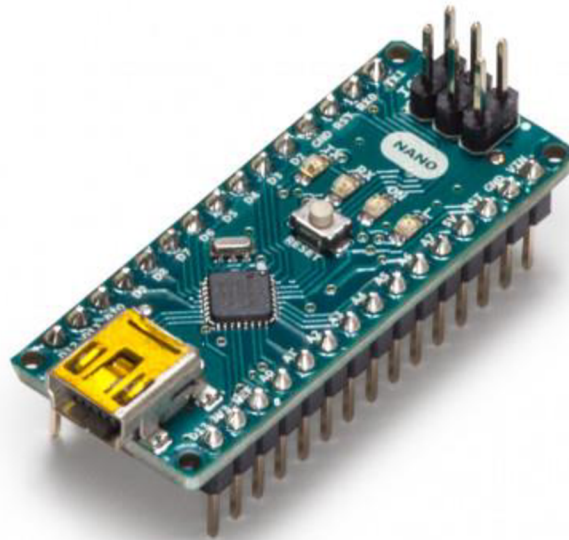- low power consumption: 19mA which operates at 5V.[15]

figure 3.1 Arduino Nano board [15]

3.1.2.Chosen Module

In the beginning of this project separate modules for GPS and GSM were used, but later continued with one module that has both features which is the Sim808. So, in this section each will be viewed briefly.

Neo7m GPS module

The first module that was started with. This module sent the data in many types which are not useful, but it was possible to pick its main important data using Arduino open source library called (Tiny GPS), in the figure 3.2 we can see the longitude and latitude position data, and the date and time extracted from this module.

```
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231189,16.570659   Date/Time: 2/11/2019 17:43:58.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
Location: 49.231197,16.570657   Date/Time: 2/11/2019 17:43:59.00
```

figure 3.2 Picked position and time data.

Sim900 GSM/GPRS module

This module can maintain internet connection for sending of data over the internet using the mobile network, as well as other functions like sending messages and making voice calls. To test its functionality, it was connected to Arduino UNO board and sent the temperature data from a thermometer to ThingSpeak IoT platform. The sending was successful and could maintain a regular sending of temperature sensor data as shown in the figure 3.3
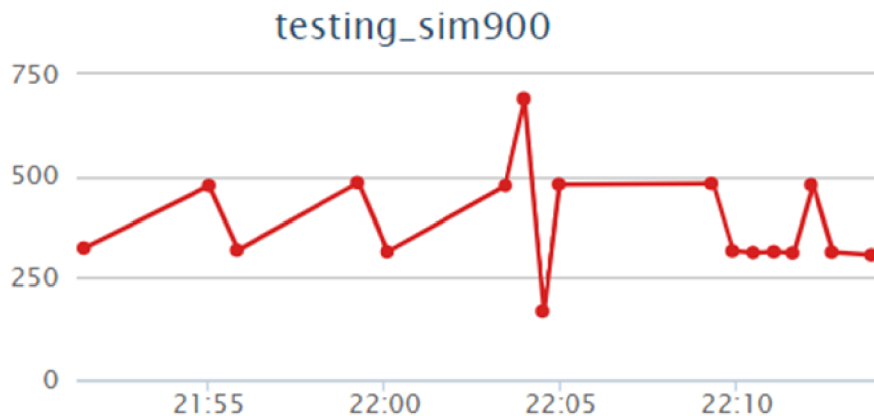

Figure 3.3: Stored sensor data received from Sim900 on the ThingSpeak IoT platform

It was noticeable that this GSM module draws a lot of power, and it will not be able to run with the Arduino board power supply alone. So external power must be insured.

Sim808 GSM/GPRS and GPS module

During the testing of the previous modules it was evident that using multiple modules with the microcontroller could cause a lot of instability issues, due to the serial line being busy. So, the best solution to that was this module that is able to do both functions.

To start with this module external power like a battery or charger was needed, and a USB to serial TTL (FTDI FT232R)for the communication between the module and the computer. Serial monitor software is needed to be installed on the computer, for this project *Termite* (a serial monitor software) was used. After connecting to the USB and opening the serial monitor program, the baud rate (9600 was set this module) and the used USB port should be selected. after this it is important to make sure everything is connected and working properly which is done by using the AT commands. To check the connection the "AT" AT command is used to makes sure the connection has been set successfully, the module will reply with "OK" if the connection is set.

The other AT commands for making sure the other features in working order will be described in the section of AT commands on chapter 3.2.

During the testing of this module it was noticed that the module shuts down after starting the GSM feature, this was due to high current consumption at the start which could not be reached by the battery, to solve this issue a polar capacitor (220 micro-farads)was added to the battery, this successfully solved the problem.

The Sim808 allows for charging the battery and once the battery is fully charged the module automatically cuts the power supply.

3.1.3. Connection Schematics and Electrical Components.

In this section we will investigate the final circuit design and describe each part of it
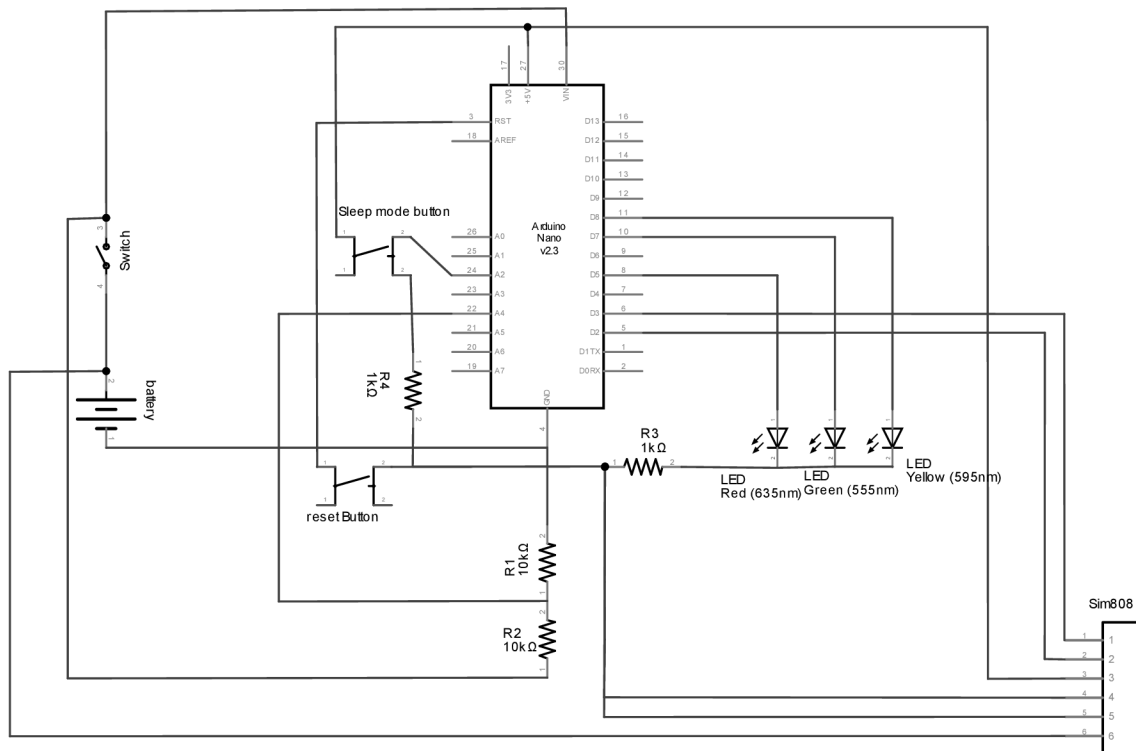

Figure 3.4 schematics of the whole device made by Fritzing.

Notice: The numbers on the Sim808 in figure 3.4 on the schematics are (TX, RX, 5V, GND, Bat GND, Bat Vin) respectively from 1 to 5

The Arduino board is the main part, and everything is connected to it, so we will look into each part and describe its function and connection with the Arduino board.

Sim808:

it has 4 pins to connect with Arduino, and 2 pins for connecting with the battery.

- TX (transmission serial pin) is connected to the Arduino digital pin D3 which is set by the code made to be the receiving serial pin of the Arduino.

- RX (Receiving serial pin) is connected to the Arduino digital pin D2, which is set by the code to be the Transmission serial pin of the Arduino.

- GND is the ground pin and is connected to the ground pin of the Arduino board.

- 5V pin is connected to the 5V output of the Arduino.

- Bat GND is connected to the ground of the battery (the negative terminal).

- Bat Vin is connected to the battery positive terminal.

Voltage divider:

It is a circuit that is made for distributing the input voltage into sections based on the components of the divider, in this project the simplest type of voltage divider was used, which is two resistors of same resistance value (10k ohm), connected in series to divide the battery voltage into two equal parts so the Arduino can read its voltage.

The Analog pins of the Arduino are capable of reading voltage, which enables them to read sensor data, but each Arduino can read a limited voltage range from 0 up to 3.3V or 5V depending on its type. For Arduino Nano that is powered by battery the range is 0 to 3.3V, meanwhile the used battery gets up to 4.2V so, a voltage divider is needed to measure the battery voltage because 4.2V is out of the range.[24]

The voltage divider on the schematics on figure 4.1.5 is the R1 and R2 resistors connected in series to the positive and negative terminals of the battery, so the voltage reading of each of the resistors is half the real value of voltage of the battery. The voltage measured here is the voltage at the resistor R1 because the Analog pin reads with reference to ground. The used pin here is A4 on the Arduino board.

Vout=Vin*(R1/(R1+R2))

Buttons:

The design of this device has two external push buttons, as well as the on-board power key of the Sim808.

Sleep mode button:

This button is responsible for the low power consumption of the device, and to stop the sending of the GPS data without totally shutting the device down. One side of the push button is connected to the 5V output of the Arduino, the other side is connected to both Analog pin A2 and a resistor in series with ground. This configuration allows the Analog pin to read the voltage. when the button is pushed the Analog will read the voltage at the resistor which is the only component so it voltage will be 5V. A 3.3V reading or more means high, less than 3.3V means low. Depending on the voltage reading (high or low) the Arduino understands if the button has been clicked.

Reset button:

This button has exactly the same function as the Arduino on board button, but due to the fact that the Arduino reset button is too small and most likely will not be exposed, an external reset button was added for better final design. This button offers the reset of the whole program and starting from beginning which is proved to help with this project (more on that on coding section).

To reset the Arduino the reset pin on Arduino should be connected to the ground. We can see on the schematics that a push button was connected between Reset pin on Arduino board and the ground.

LEDs:

As we can see in the schematics a set of three LEDs (light emitting diode) were used. Each LED is connected to one digital pin, and on the other side they all are connected in series to a resistor to the ground. the resistor function here is to reduce the amount of current passing through the LEDs. without the resistor the LEDs might burn and permanently stop functioning. A 1k ohm resistor was used. After experimenting with other resistors, the 1k ohm was bright enough yet have a lower power consumption than lower resistor values.

Battery and switch:

The battery positive terminal was connected to the power switch and the Sim808 input BatVin (to be able to charge the battery without the need to switch on the device). And the negative connected to the shared ground of all device.

The switch connects the battery to the Arduino input pin Vin and to the voltage divider.

3.1.4. Power Consumption and Battery

To get a good idea how much power the final device consume we need to look at the documentation, but also real life measurement are essential because usually the numbers giving from the manufacturer of these chips are at certain mode, which might be changed for the purpose of this project. So, we will investigate the energy consumption from documentations and tests in this section.

The unit for measuring power consumption is Watt, but due to the fact that the voltage is constant (5V for this module/ also USB offers 5V supply) many documentations shows the power in terms of current consumption in [mA]. Another unit need to be understood is the [mAh], this is the unit usually used to identify battery power capacity. A 1mAh describes the ability of a battery to serve 1mA for 1 hour time interval.

documentation numbers:

Sim808:

The documentation power consumption was mentioned in the chapter 2.2.4 on the topic of Sim808 power consumption.

Arduino Nano:

The official website says 19mA, but that would vary depending on the kind of usage.

Electrical components:

LEDs are made to blink for a fraction of a second instead of constant lightning to save on power. So, the major consumer of power of all component other than the Arduino and the Sim module is the voltage divider.

and the power consumption is the current times the voltage. the total resistance is 20K ohm which leads to a current of

$I = U/R = 4/20000 = 0.0002$ A $= 0.2$ mA (assuming that battery voltage is constant at 4V)

leading to a power consumption of:

$P = U * I = 4 * 0.0002 = 0.0008$ Watt $= 0.8$ mW

Real power testing:

The following numbers were measured while the GPS device was sending the GPS data regularly, which is the normal working mode. During this process, the current (and the power consumption) was alternating up and down depending on the process being taken at that moment (GPS fixing, sending data over network, waiting ...etc).

So the best way to measure power consumption was by using a USB metre which is a device used for measuring the power consumption of USB connected devices. This device allows for measuring current consumption over time in [mAh], and power consumption over time in [mWh]. For example, it can measure the whole consumption of 10 minutes of running the device. These tests were made sections of 10 minutes of running the device for a high accuracy measuring.

Sim808: 60 mA (both GPS and GSM engines were working)

which translates to power of P=U*I = 5 * 0.060 = 0.3 Watt = 300mW

Arduino: 22.5 mA (during connection to computer)

Which translates to power of P=U*I = 5 * 0.0225 = 0.113 Watt = 113mW

The Arduino and the Sim808 together: 72.2 mA

This result is lower than the sum of each of them alone but that was due to the fact the Arduino measuring was while it was connected to the computer which adds the consumption of the USB chip on the Arduino, based on documentation it consumes 15mA on normal operation. [25] page 15

Adding the voltage divider gets to 72.4 mA

Calculating the power, we will get P = U * I = 5 * 72.4 = 0.362 Watt = 362 mW

Battery:

During this project and testing outside, the whole assembly was powered by battery, two types were tested

The first was the 420 mAh - 3.7V small rechargeable battery which is shown on the figure 3.5. It has a small size of 40x50x4 mm. 420 mAh mean it can supply up to 420mA for one hour, but these numbers usually are more than the reality, and get reduced with time as the battery cycles being used. 3.7V means that the battery mainly supplies voltages around this value more or less.

In theory this battery should last:

T = battery [mAh] / device consumption [mA] = 420 / 72.4 = 5.8 hours

But this assumption is usually incorrect due to the fact that battery capacity is at certain conditions, and as the battery being used its capacity gets reduced.

figure 3.5 420mAh Li-ion battery

The second battery was the 3800 mAh - 3.7V bigger battery. It has a size of 40x50x6mm which is shown on the figure 3.6.

Figure 3.6 3800 mAh Li-ion battery

The smaller battery was the first choice to help reduce the overall size of the final device, but due to the instability of the working of the device specially at starting while using the smaller battery when it was not fully charged. It would sometimes not be enough to get the GPS fix which require a high current at the start up even that it can start the device. It could only work properly when the small battery was fully charged but after around one hour of usage, the smaller battery could not supply enough current. So, the larger battery of 3800 mAh was used in the end to offer a higher level or reliability to final device.

Battery capacity measuring:

It can be hard to measure the left capacity of a battery (the battery life) accurately, the only accurate way is to drain all the capacity and measure how many mAh was drained, but even this can give different results depending on the discharging current amount. usually manufacturers set the value of current at which the battery would give the nominal capacity.

To measure the left capacity of the battery there are multiple methods, one of them is by relating to the voltage, this works because voltage decreases as the battery being used. also, with the device configuration we have it was easy to measure the voltage using the Arduino Analog pins. It is important to note that this method is not accurate but it was simple to use in this project.

to have a good insight about the voltage capacity relation we can look at this graph in figure 4.1.8

Before that we need to understand the C-rating:

C-rate is a measure of the rate at which a battery is discharged relative to its maximum capacity. A 1C rate means that the discharge current will discharge the entire battery in 1 hour. For a battery with a capacity of 420 mAh, this equates to a discharge current of 420 mA. A 2C rate for this battery would be 840 mA, and a 0.5C rate would be 210 mA.[26]
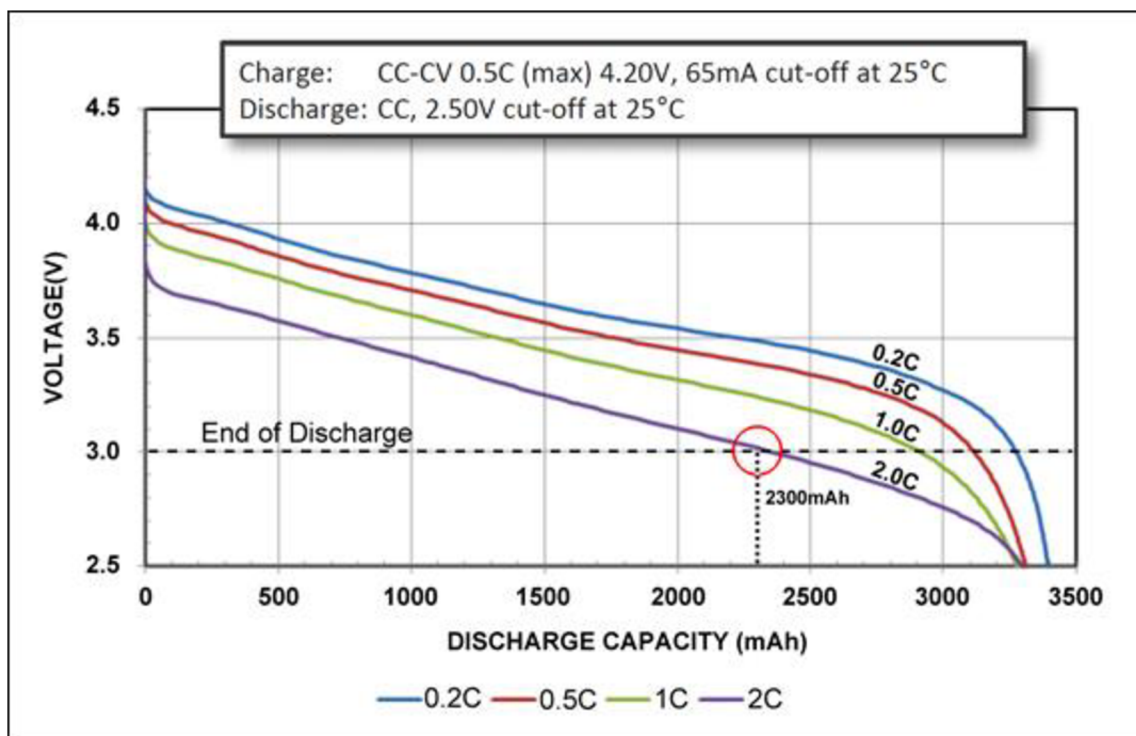

figure 3.7 Discharge characteristics of NCR18650B Energy 3200mAh Cell by Panasonic.[40]

In figure 3.7 we can see the difference between discharge rates related to amount of discharge current. Also, we can see that voltage decreases as the capacity is being used. We can assume that the change is linear to give an approximation for the capacity of the battery at a given voltage.

Li-ion Batteries have similar discharge curves and the one in the figure 4.1.8 describes how the discharge curve should look like, this graph was chosen due to lack of datasheet of the used battery. The used battery (*Ultrafire* 3800mAh)has range of voltage of 2.75 – 4.2V (cut-off voltage – full charged).

For estimating the capacity, the real range of working voltage should be 3.1 – 4.1 V, leaving us with 1V range difference. For estimation we can relate the amount of voltage to the battery left capacity. By experimenting it was evident that the voltage of 4V is the most stable stage then battery starts to lower fast once it reaches 3.7V. So it was chosen to keep the Voltage reading as the main battery estimation method rather than converting it to a percentage.

3.1.5. Protection of the device

This GPS device was made to be used outside for tracking purpose which requires a box with proper design to save it from mechanical damage, and to make sure all components are within one as small as possible box.

One of the possibilities of the design was to make the box by a 3D-printer which allows for highly configurable design, yet strong material made with one single body, which is stronger than multiple parts connected together. So 3-D printed box was the choice to save on size and provide more protection.

Design requirements:

Before the design, some essential ideas should be set to suit the design with the component needs. based on these ideas the design was guided:

- Must have openings for charging of the battery, Sim card changing, and Arduino USB for enabling changing of setting of the device, as well as the opening for the control panel (buttons, switch and LEDs)

- As tight as possible to lower the size as much as possible.

- Removable cover to allow for some fixing if needed (favourable if can be tightened with screws)

- The Sim module should have an exposed power key. This was due to the way Sim808 model we had (model number : CT0078SIM). This model board design give access to communication serial pins and ground and 5V, as well as the battery intakes, but does not give access to the Power key, so the on board sim808 power key must stay exposed to be used when needed.

With these ideas in mind the design was made using the designing software *Autodesk Inventor*, the final design came to be as figure 3.8. The design parts were converted to the file type (.stl) for the "Ultimaker" 3D-printer used.

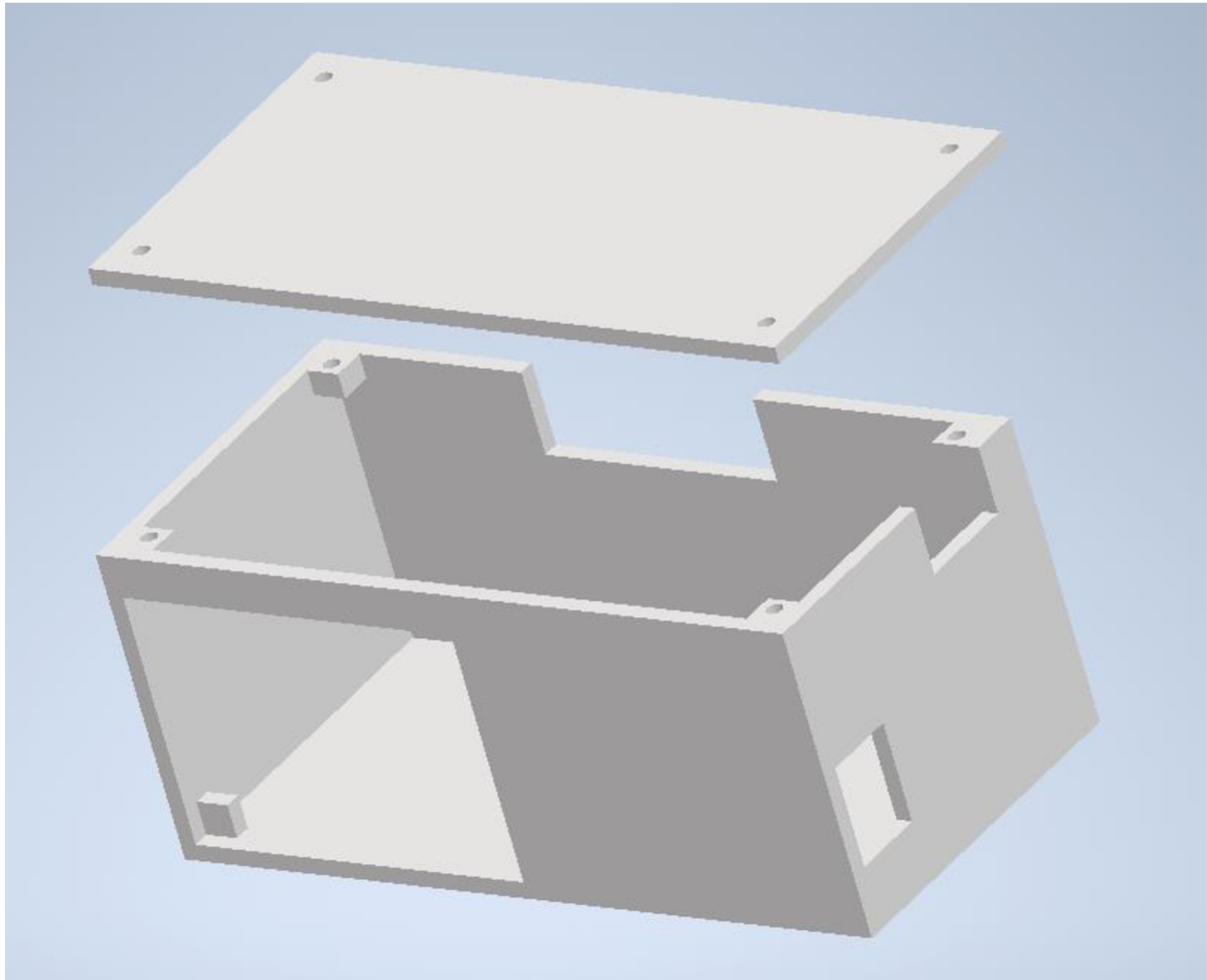The overall design of the exterior measurements came to be 88x43x39 mm, with wall thickness of 2 mm.

figure 3.8 box design for the GPS device using Autodesk Inventor

The material of choice was the ABS (Acrylonitrile butadiene styrene) due to its strength. ABS Typically ranks as the second most popular 3D printer filament, after PLA. With respect to ABS material properties, it is actually superior to PLA, despite being slightly more difficult to print with. It is for this reason that ABS is found in many manufactured household and consumer goods, including LEGO bricks and bicycle helmets. Some of the properties are High strength, high durability, and resistant to high temperatures. [27] The finished product design is shown in figure 3.9
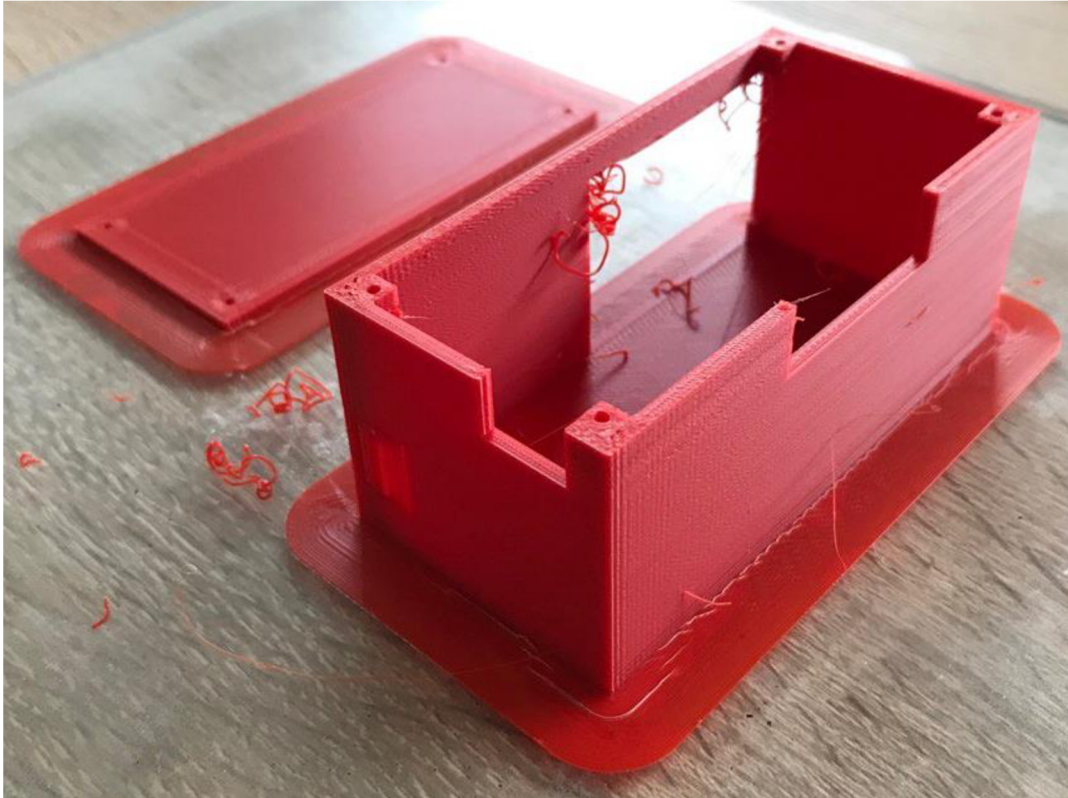
figure 3.9 the finished printed design of the GPS device box

Hot glue was applied to make sure all the components would stay in their place safely, hot glue was the option since its relatively easy to remove in case of some bigger changes to the design in the future, yet hot glue is reliable to keep everything in its place. The figure 3.10. shows the components inside the box in their place. And figure 3.11 shows the final shape of box with component within.
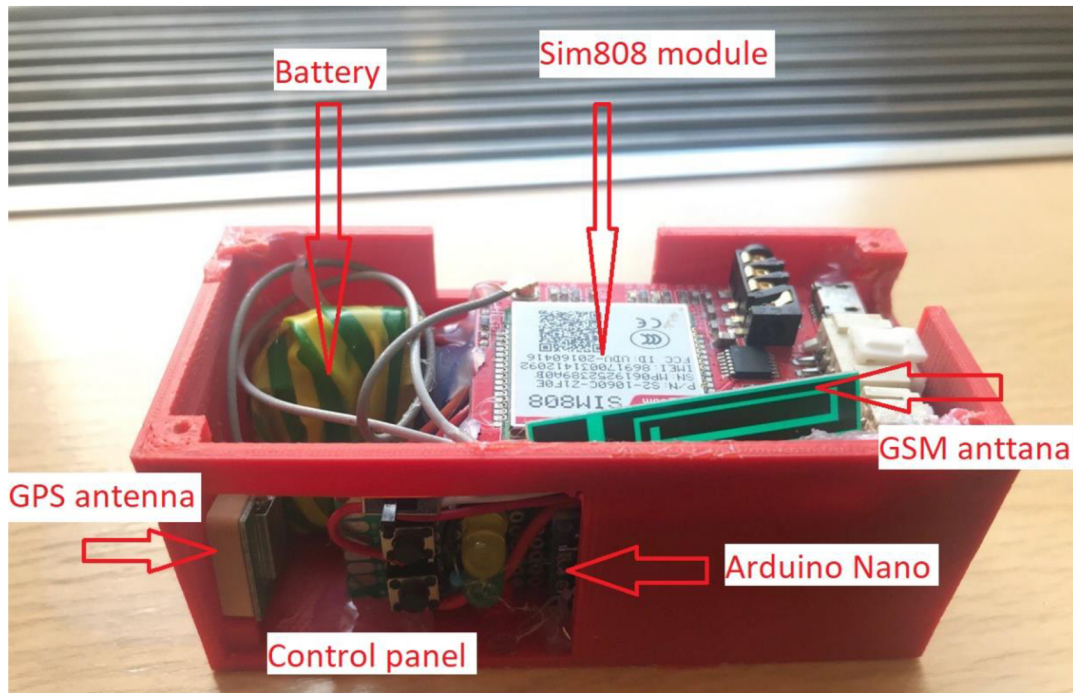
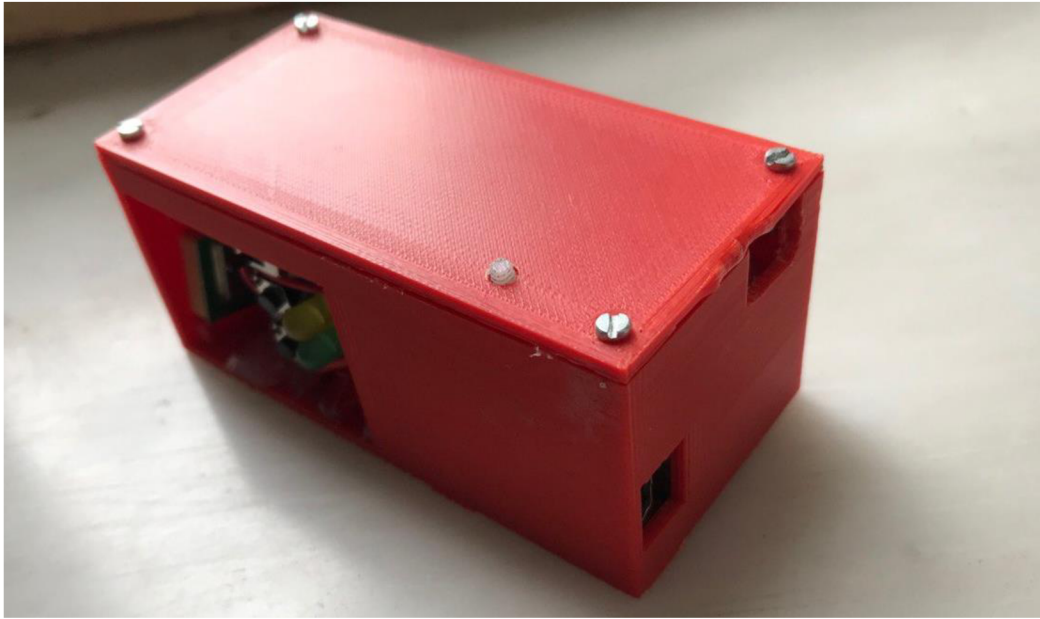Figure 3.10 The layout of the components inside the box.

figure 3.11 final shape of the device

The main drawback to the box design is the squareness feel, if the design would be remade round edges would be the choice.

3.2. Code and functionality

In this section we will look into the coding and communication between the microcontroller and the module, and whole program used for the Arduino in general to get a clear idea how it works. First we will look into AT commands as they are the language that Sim808 understand then to the Arduino code.

3.2.1. AT Commands

AT Commands are commands used to control the GSM moulds and modems via its serial interface to use their functionality of sending messages, voice calls, and data transfer over the mobile network. It is important to note that every GSM module can support a limited AT Commands set for that GSM module, and these commands get updated for newer versions of hardware. In this project the Sim808 GSM module was used, which belongs to the Sim800 series of AT commands.

Another feature of the Sim808 AT Commands is the AT commands for the GPS engine is within, So the whole Sim808 is controlled by the AT commands GSM and GPS functions.

AT Commands have distinctive syntax usually comes in the form "AT+SOMETHING", to set a value with for that command "AT+SOMETHING=<FirstParameterValue>,<SecondParameterValue>", to ask about the current value set for a command "AT+SOMETHING?". Here are some of the used AT commands and their roles described briefly. The AT commands in the following table are in the sequence of working of the final program:

AT Command	Description	Response
AT	Is the command for making sure the serial connection is maintained, it also sets the baud rate automatically if it was set to automatic. the expected response is "OK", not receiving OK means serial connections was not maintained.	OK
AT+CPOWD	Power off, turns off the whole module AT+CPOWD=0 Power off urgently AT+CPOWD=1 Normal power off (used)	[NORMAL POWER DOWN]
AT+SAPBR	Used for Bearer Settings for Applications Based on IP, consists of 4 values in the syntax: AT+SAPBR=<cmd_type>,<cid>[,<ConParamTag>,<ConParamValue>] <cmd_type> 0 Close bearer 1 Open bearer (used) 2 Query bearer 3 Set bearer parameters (used) 4 Get bearer parameters <cid> 0 Bearer is connecting 1 Bearer is connected (used)	OK ERROR

	2 Bearer is closing 3 Bearer is closed <ConParamTag> Bearer parameter "CONTYPE" Type of Internet connection. "APN" Access point name string: maximum 64 characters (used) "USER" User name string: maximum 32 characters "PWD" Password string: maximum 32 characters "PHONENUM" Phone number for CSD call "RATE" CSD connection rate <ConParamValue> Bearer parameter value	
AT+CGNSPWR	This command returns the status of GNSS engine power supply, its value can be set to: 1 turned on (by typing the command "AT+CGNSPWR=1" 0 turned off	1 0
AT+CGNSINF	GNSS navigation information parsed from NMEA sentences. These are the GPS data we need. The response comes in this syntax: +CGNSINF: <GNSS run status>,<Fix status>, <UTC date & Time>,<Latitude>,<Longitude>, <MSL Altitude>,<Speed Over Ground>, <Course Over Ground>, <Fix Mode>,<Reserved1>,<HDOP>,<PDOP>, <VDOP>,<Reserved2>,<GNSS Satellites in View>, <GNSS Satellites Used>,<GLONASS Satellites Used>,<Reserved3>,<C/N0 max>,<HPA>,<VPA> OK Notes: the values come in this form for: Fix status (0-1) UTC date & time (yyyyMMddhh mmss.sss) Speed over ground (Km/hour [0,999.99]) More on this command on the documentations [28]	
AT+HTTPINIT	Initialize HTTP Service. This command should be sent before starting HTTP service.	OK
AT+HTTPPARA	Set HTTP Parameters Value. This command allow for saving the parameters of the final URL to be sent later to the website by another AT command. It has the syntax of: AT+HTTPPARA=<HTTPParamTag>,<HTTPParamValue> <HTTPParamTag> is the type of parameters set. can be: "URL" ,"UA","CID"…..etc The important to us is "CID" and "URL" "URL" : (Mandatory Parameter) HTTP client URL has the form "http://'server'/'path':'tcpPort' " "server": FQDN (domain name) or IP-address "path": path of file or directory (name and each value of the sent data)	+HTTPPARA: "HTTPParam Tag", "HTTPParm Value"

	"tcpPort": default value is 80. "CID" (Mandatory Parameter) Bearer profile identifier (identified as 1)	
AT+HTTPACTI ON	HTTP Method Action. it is used to perform HTTP action such as HTTP GET or HTTP POST **AT+HTTPACTION=** **0** HTTP GET(used to request data from a server, and also the query string (name/value) is sent in the URL of the GET request, this query string have all the sent data.) (used) **1** HTTP POST(used to send data to a server) **2** HEAD	
AT+HTTPREAD	Read the HTTP Server Response. Read all data when AT+HTTPACTION=0 or AT+HTTPDATA is executed	+HTTPREAD: <date_len> <data> OK

[28][30]

a sequence of these AT commands would allow for receiving GPS data and sending it over the network. but a very important notice is that these commands cant be sent to the Sim module directly one by one fast without letting the processor finish each of them and reply to it, So enough time should be given and reply should be detected for the last AT command before sending the next one.

3.2.2. Code and Algorithm

In this part we will have a look at the code of the Arduino microcontroller that is receiving, identifying, analysing, and sending of the GPS data. The code that was used is an open sourced code from a blogger/youtuber knows as "iforce2d", the source code can be found at [29]. The code was modified for the needs of this project.

This code is written in the Arduino programming language, which is a combination of C and C++ functions. this code resembles the AT command sequence needed to power the Sim808 module, receive GPS data, opening the network communications, and sending the data to the server (the website). Now let us look at the main pieces of the code and their functions:

library:

Only the default library SoftwareSerial was used. The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino.

#include <SoftwareSerial.h>

serial communication

The code starts with setting serial communication, the serial pins, and the Baud rate (the speed at which data are transferred)

SoftwareSerial ss(2,3); → set digital pins 2 and 3 as serial pins between Arduino and Sim808.

ss.begin(9600);   → setting baud rate at 9600 between Arduino and Sim808.

Serial.begin(115200);   → setting baudrate at 115200 between the computer and Arduino.

The function responsible for sending AT commands:

once we send the Sim808 an AT command we need to understand that we should not sent another command after unless the Sim808 responds, otherwise some commands will not be executed, to make sure that we received a response we need to use a function as follows:

void sendGSM(const char* msg, int waitMs = 500) {

 GSM_PORT.println(msg);

 while(GSM_PORT.available()) {

 parseATText(GSM_PORT.read());}

 delay(waitMs);}

This function will pass the AT command we want to send to the Sim808, with the estimated time delay afterwards that will help making sure the command was executed, after that the response message is taken for the next step by the function "parseATText()"

Main idea:

The structure behind the code is that the code is split into multiple states, each state describes the stage which we are at, to be able to name different states of the process the function "enum" was used to allow us to define a set of named constants. Using Enums can make it easier to create a set of distinct cases.

Once the Arduino receive the response back, the response is directed to the function "parseATText()" which will send it to a switch function to the first case "PS_DETECT_MSG_TYPE", which will read the response to know what stage we are at, by comparing the response to a set of expected responses, using strcmp() function, if it matches an expected response we know what stage we are at and proceed in that stage (case).

Examples:

If the response starts with "AT+" we know this is only an echo of the sent AT command, so it will be directed to the case "PS_IGNORING_COMMAND_ECHO".

If the response is "+CGNSINF:" we know this is the value of detected GPS position, so it will be directed to the case "PS_CGNSINF_RUN_STATUS". this case will extract the desired GPS data from the string.

Next:

Once fix status equals to 1 and all GPS values are set, these values will be set as the parameters of the "AT+HTTPPARA" which is the parameters saving AT command, then "AT+HTTPACTION=0" is used to send these parameters to the server via HTTP GET URL.

Once the server response is received a LED the green LED will blink, indicating a successful sending of data.

LEDs:

To identify the LEDs to the Arduino, we need to set the connected digital pin of each LED as an output using this code:

int GPSLED = 8;   → to identify the used pin (digital pin 8 in this case)

pinMode(GPSLED,OUTPUT);   → setting the pin as an output

digitalWrite(GPSLED,HIGH);   → turns on the LED

   delay(100);   → make a delay of 0.1 seconds (100 mS)(the blink period)

digitalWrite(GPSLED,LOW);   → turn off the LED

these last 3 code lines represent the blink, placing them at the corresponding case that the LED represent makes it blink every time that case is fulfilled. for the GPS LED (yellow) it blinks once the GPS fix was determined.

Button:

To make the program identify button, we need to set an Analog pin that is connected to the button as input, and there will be 2 cases only

≥ 3.3V → high (when clicked)   or   < 3.3V → low (when not clicked will be 0V)

the code is :

int standbyBTN = A2   → to identify the used pin(analog A2 for this button)

pinMode(standbyBTN,INPUT);   → setting the pin as input

if(digitalRead(standbyBTN)= =HIGH) {   → the if statement for the button to work

  delay(500);

  sendGSM("AT+CPOWD=1");

  stop(); }

This button was to set the standby mode, during testing it was found that the best way is to shut down the Sim808, while putting the Arduino to sleep by an open loop. to wake up the device the reset button should be used and the power key of the Sim808.

Battery Measuring

the Arduino can read the voltage using the Analog pins, the voltage read is based on the voltage reference of the Arduino board some have 3.3V and some have 5V. the Arduino Nano showed 5V charged from the USB, and 3.3V charged from the battery. meanwhile 3.3V is less than the battery voltage so a voltage divider (reads half of the real voltage).

The reading from the Arduino can be from 0 to 1024 units, of same proportion from 0 to the reference voltage. So, 3.3/1024 = 0.00322 V/Unit [24], with this in mind the following code was constructed:

int analogPin = A4;   → to identify the used pin(Analog A4)

bat = ((analogRead(analogPin) * 0.00322) * 2) - 0.5;

notice: the reading was multiplied by 2 due to the use of voltage divider, and minus 0.5 due to the deviation is the reading on the Analog pin (deviated by 0.5V). the deviation amount was measured using a Multimeter to measure the real voltage and the reading from Arduino.

Safety against over discharge

For the safety of the battery from over discharge the code was made to turn on an LED (low battery LED) when the battery voltage is under 3.3V to indicate low battery life using the the IF function as the following code:

if (bat < 3.3) {

    digitalWrite(LBLED,HIGH); }

For the automatic shut down of the device the AT command for power down "AT+CPOWD=1" is sent when voltage gets down to 3V using the following code:

if (bat < 3) {

    sendGSM("AT+CPOWD=1"); }

It is important to to notice that the Arduino board cant be turned off using code, so switch will be the final step to totally shut down the Arduino.

3.3. The Web Application

This part we will view the chosen application for storing and viewing of the GPS data received from the GPS device. in the chapter 2.3 we reviewed soe of possible ways for the web application. after looking into the possibilities offered by the google maps JavaScript API it was chosen for this project. The made website with map and saved log from some trips can be seen on the website URL:

https://vutmaizi.000webhostapp.com/A.php

3.3.1. Google Maps JavaScript API

The google maps JavaScript offers customizable maps to be infused in a website using the JavaScript programming language. It is powerful platform with many options offering a wide range of modification and development to the project.

To start with google maps JavaScript API, an API key must be generated. The API key is a unique identifier that is used to authenticate requests associated with each project for usage and billing purposes. The google maps java script is free to use, but in case it is to be used for business purposes the user have to pay for the services based on the traffic. The API key can easily be made from the "Google Cloud Platform Console". within the API key setting. The API key can provide a high level of security for the map, as well as limiting the usage to some websites or IP addresses.

After the API key is generated it is called inside the website as a JavaScript code, then the map is initialized, and modifications would be added.

The generated API key was: AIzaSyCg7BCoeJFaaJb1GVlaDeJ2slWhox18pXw

and is called as a JavaScript like this:

```
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCg7BCoeJFaaJb1GVlaDeJ2slWhox18pXw"></script>
```

3.3.2. Code Algorithm Overview

For the Website made three Programming language were used. The first is **HTML** mainly to organise the look of the website, and to allow for inputs, and viewing of values.

The Second is **PHP** contains the GET request to save data received and responds back.

The Third is **JavaScript**, this is the main code and algorithm, in this section the map is initialized, the map options are set, the markers are added, and lines to connect these markers.

The whole code had the extension (.php), and every language had to be identified before and after it is used.
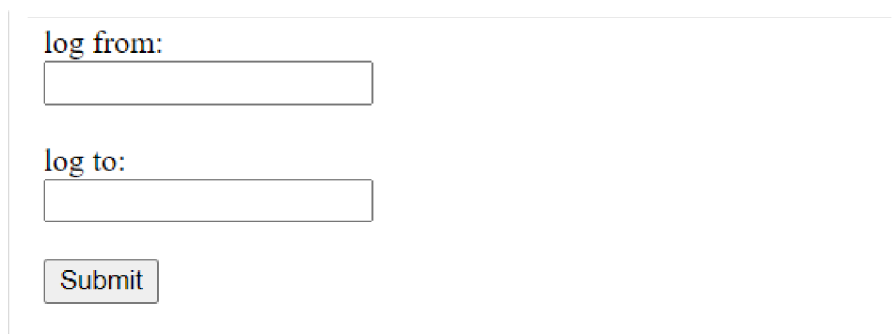
The code is divided into 2 pages(websites), the first is only for receiving the GPS data and to store it. The second page is where the map is constructed. the code for these pahes will be provided as an attachment.

HTML code:

```
<!DOCTYPE html>
<html>
<form action="/index.php">
<label for="from">log from: </b><br /></label>
<input type="number" name="from" id="from"><br><br>
<label for="to">log to:  </b><br /> </label>
<input type="number" name="to" id="to"><br><br>
<input type="submit" value="Submit">
</form>
```

figure 3.12 HTML code used to set input boxes.

The code in figure 3.12 represents a form of two input boxes. These inputs are made so the final website will be able to view a specific range of GPS log based on the total number of logs. The first box has the label "log from" and the data type is set to be a number saved as "from", second box is labelled as "log to" and the data type is set to be a number saved as "to". The resulting shape on the website looks like figure 3.13

log from:

[   ]

log to:

[   ]

[Submit]

figure 3.13 The input boxes for the wanted GPS range to be viewed.

PHP code:

As mentioned before there are two pages, PHP code is in the one responsible for saving the data. It can be found in the attachment as "saver.php"

```php
<?php

  if (!empty($_GET['latitude']) && !empty($_GET['longitude']) &&
     !empty($_GET['itime']) && !empty($_GET['satellites']) &&
     !empty($_GET['speed']) && !empty($_GET['battery'])) {

     function getParameter($par, $default = null){
       if (isset($_GET[$par]) && strlen($_GET[$par])) return $_GET[$par];
       elseif (isset($_POST[$par]) && strlen($_POST[$par]))
         return $_POST[$par];
       else return $default;
     }

     $file = 'gpsdata.txt';
     $lat = getParameter("latitude");
     $lon = getParameter("longitude");
     $itime = getParameter("itime");
     $sat = getParameter("satellites");
     $speed = getParameter("speed");
     $battery = getParameter("battery");
     $person = $lat.",".$lon.",".$itime.",".$sat.",".$speed.",".$battery."\n";

     echo "
       Great";

     if (!file_put_contents($file, $person, FILE_APPEND | LOCK_EX))
       echo "\n\t Error saving Data\n";
     else echo "\n\t Data Save\n";
  }

?>
```

Figure 3.14 PHP code of the saver.php file.

The PHP code in the figure 3.14 uses the following function to receive the data from the URL sent by the GPS device

$_GET['name']   → this name should be identical to names of the query string sent.

If the full set of names and values have been received the code will save them in a text file (.txt) in lines using "file_put_contents" PHP function, each line represent one log of GPS data in the code the line of values named as "$Person" (a variable), these data in the following arrangement with commas in between:

Latitude,Longitude,date&time,SatelliteSignal,Speed,battery

If the data was successfully saved, this PHP page will send a message to the GPS device "Data Save", if not the message will be "Error saving Data".

JavaScript code:

The JavaScript code starts with the < script > identifiers for the PHP server to understand the following code is a JavaScript.

Initializing:

```html
<script type="text/javascript" src="jquery-1.11.2.min.js"></script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCg7BCoeJFaaJb1GVlaDeJ2slWhox18pXw">
</script>
<script type="text/javascript">
  var myCenter=new google.maps.LatLng(49.231138,16.570412);
  var marker;
  var map;
  var mapProp;
  var flightPlanCoordinates = [];
  var ffrom="<?php echo $fromlog; ?>" || 0;
  var tto="<?php echo $tolog; ?>";

  function initialize()
  {
    mapProp = {
      center:myCenter,
      zoom:18,
        mapTypeId: 'satellite'
      };
    mark();
```

figure 3.15 The code for initializing google map.

The code starts with identifying the library used "jquery-1.11.2.min.js". Then setting the map API key. Then the used variables are defined.

It was not possible to call a variable in the HTML part of the code to the JavaScript part, so PHP was used to move the input values. This was made by using the PHP $_GET function to take the values from the HTML part as the code in figure 3.16

```php
11    <?php
12    $fromlog = $_GET['from'];
13    $tolog = $_GET['to'];
14    ?>
```

figure 3.16 PHP code

47

Then in the JavaScript part the PHP function "echo" was used to output these values. these values names in the JavaScript are "ffrom" and "tto", the "||" logical operator was used to set the "ffrom" value to zero if there was no input which helps make the usability of the website easier.

After that, the map initialization function is called, within this function are some of the map properties, after setting them the function of markers and lines is called.

```javascript
function mark()
{
  var latlon;
    var file = "gpsdata.txt";
    $.get(file, function(txt) {
        var lines = txt.split("\n");
    for (var i=ffrom;i<tto;i++){
      console.log(lines[i]);
      var words=lines[i].split(",");

      if ((words[0]!="")&&(words[1]!=""))
      {

          var titl = ["Date",words[2].slice(0,4),words[2].slice(4,6),words[2].slice(6,8),'
          var t = titl.toString();
          var currenttime = [words[2].slice(0,4),words[2].slice(4,6),words[2].slice(6,8),'
          marker=new google.maps.Marker({
          position:new google.maps.LatLng(words[0],words[1]),
          title: t
            });
      marker.setMap(map);
      map.setCenter(new google.maps.LatLng(words[0],words[1]));
      document.getElementById('itime').innerHTML=currenttime;
      document.getElementById('sat').innerHTML=words[3];
      document.getElementById('speed').innerHTML=words[4];
      document.getElementById('battery').innerHTML=words[5]; //lines.length
      document.getElementById('logs').innerHTML=lines.length;

      latlon = new google.maps.LatLng(words[0],words[1])
      tripPath.getPath().push(latlon);


      }
    }
}
```

Figure 3.17 The code for markers

The code in figure 3.17 represents the function that firstly calls the text file where all the logs of GPS data are saved, then defining lines to be an array of lines split by the (return). after that the for loop takes every line within the range and extracts the values split by commas "words[]". after each line, a marker is made using the latitude and longitude with the "marker.setMap(map);"

 as well as the title for each mark which have extra information (Date, time, speed, and signal quality) at that position.

for each line, an object "latlon" is being constructed to store latitude and longitude pairs, these objects are injected to array called "tripPath" using the function "getpath().push()", this array will be used to construct the line between markers.[31]

After the for loop reaches the end of the range the last marker is set with bouncing effect to identify the end of that range or trip.

```
72              }
73              marker.setAnimation(google.maps.Animation.BOUNCE);
74          });
75          map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
76          var tripPath = new google.maps.Polyline({
77              path: flightPlanCoordinates,
78              geodesic: true,
79              strokeColor: '#FF0000',
80              strokeOpacity: 1.0,
81              strokeWeight: 2
82          });
83          tripPath.setMap(map);
84      }
85  }
86      google.maps.event.addDomListener(window, 'load', initialize);
87  </script>
88  </head>
```

figure 3.18 setting polylines

Using the array "tripPath" mentioned before and setting the line properties, the lines connecting the markers are made by the code in figure 3.18.

3.3.3. Host server

To be able to make a website, a web hosting service is needed to allow for accessibility of the website. During tests a local website is enough for testing but it will only allow access of the Website made on the same computer.

The used local wen host software was the "WampServer" which allow access of the saved websites in special file, and can be accessed using the URL:

http://localhost/Name_of_php_file.php

But to make a website accessable from the internet by other computers or phones, there must be a host server. In this project the web hosting free sercive from *www.000webhost.com/* was used. it was effeciet and free service.

On the files section, all the needed files for the website to function are saved, including the data base and main PHP files, as well as some needed libraries.

3.3.4. Sample

The end result of the website looks like figure 3.19, this view of the website is not focused on the map but the whole structure, in the next chapter more images will focus the map.

The input boxes identify the log range from-to.

The information table above the map is data from the last log in the range, this was mainly made for battery reading during the real-time GPS tracking, but it was useful to add more data than just the battery. The choice of battery reading was kept in voltages to keep track of accurate battery voltage situation which might not be the case if it would be in percentages.
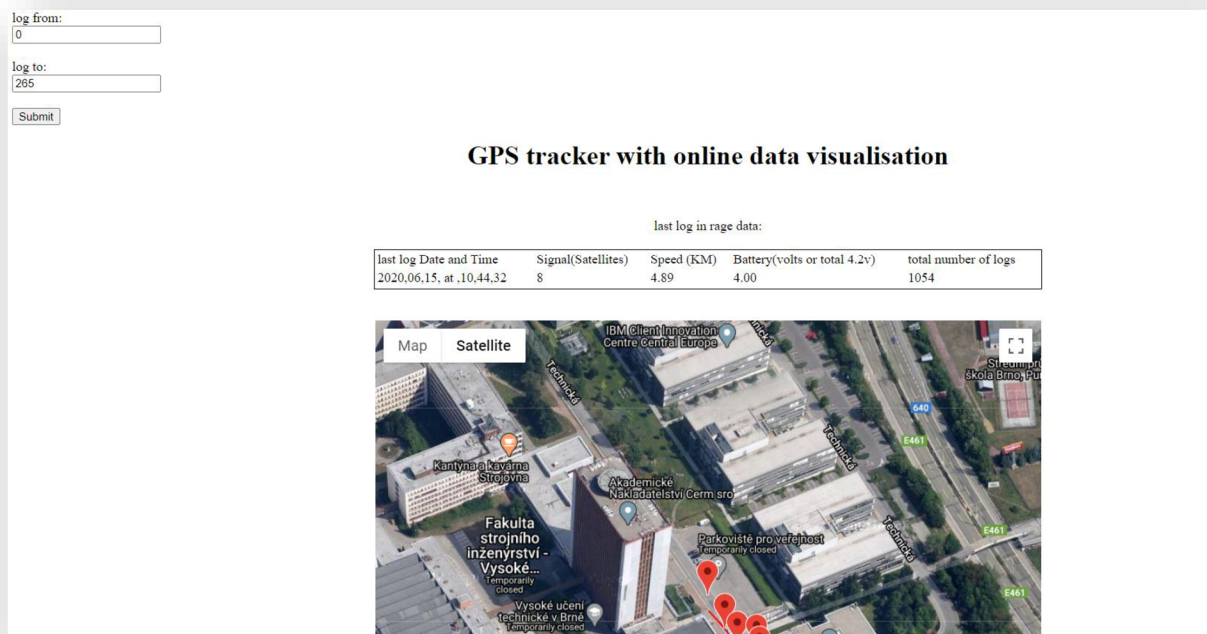

Figure 3.19 The made website

4. Testing and results

4.1. Antenna and signal stability

The GPS antenna is an essential part for receiving the GPS data from the satellites, and the position of the antenna is quite important, due to the satellites GPS signals being weak and easily blocked by objects, That is why we see cars GPS antennas placed exposed on the top of cars. During tests, sometimes it is not be possible to get GPS fix inside the room and the only way would be to place the antenna (whole device) near the window to be able to get enough satellites signal to get GPS fix.

Types of Antennas:

During this project two types of antennas were used:

Passive Patch Antenna:

The most common antenna type for GNSS applications is the patch antenna. Patch antennas are flat, generally have a ceramic and metal body and are mounted on a metal base plate as viewed on figure 4.1.
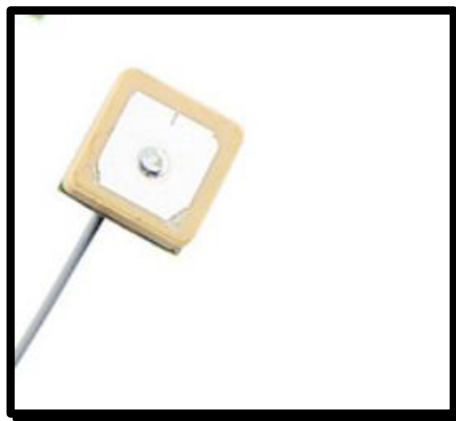
Figure 4.1 passive patch antenna [36]

Active Patch antenna:

Is a passive antenna that has an on-board low noise Amplifier. Is shown in figure 4.2

figure 4.2 active antenna [37]

After using both antennas, the difference was barely noticed in many different conditions, the main difference is maybe one extra more satellite received. The important feature of the active antenna is it allows for longer wiring which in many cases help with mounting the antenna on a better receiving position (open space). We can see the difference in both signal and accuracy for both antennas in figure 4.3. This was measured in an open space area and we can notice that active antenna has a slightly better signal and accuracy on the road.

Figure 4.3 passive (left) VS active (right) antennas

Signal stability

the GPS device maintained a stable GPS positioning in all conditions and environments, but the accuracy vary a lot depending on surroundings as we can notice in figure 5.1.5 there was a very low accuracy the estimated error reached up to 100 metres when inside buildings, due to GPS satellites signals being weak once they reach the earth and they can be blocked by objects.

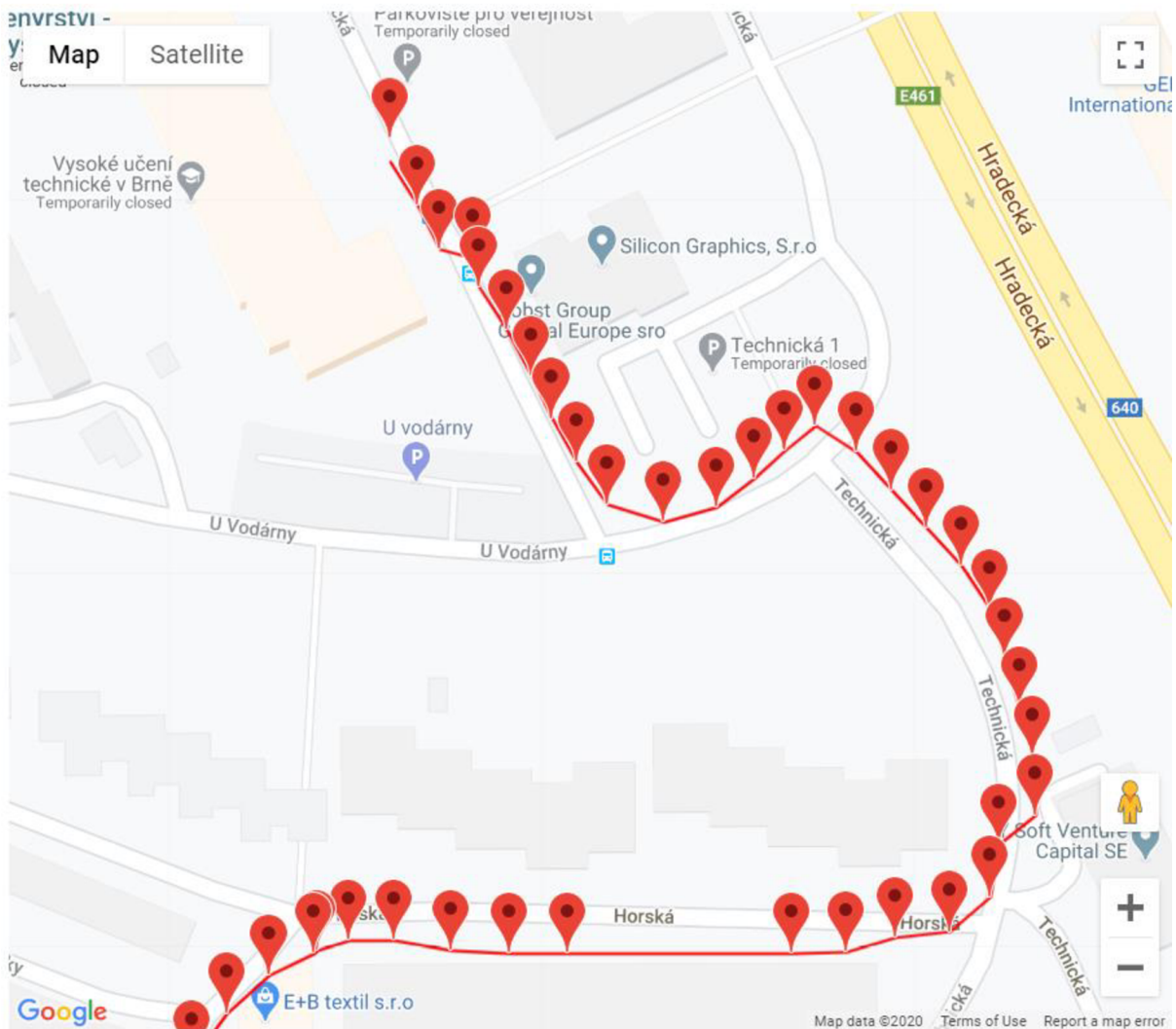The GPS device was tested in different surroundings as follows:

Urban places:

Streets:

Figure 4.4 Street tracking sample.

From the figure 4.4, we can see that accuracy is deviated by 1-3 metres in most cases (own estimation based on knowing the area). Meanwhile the track is reasonably accurate and useful.
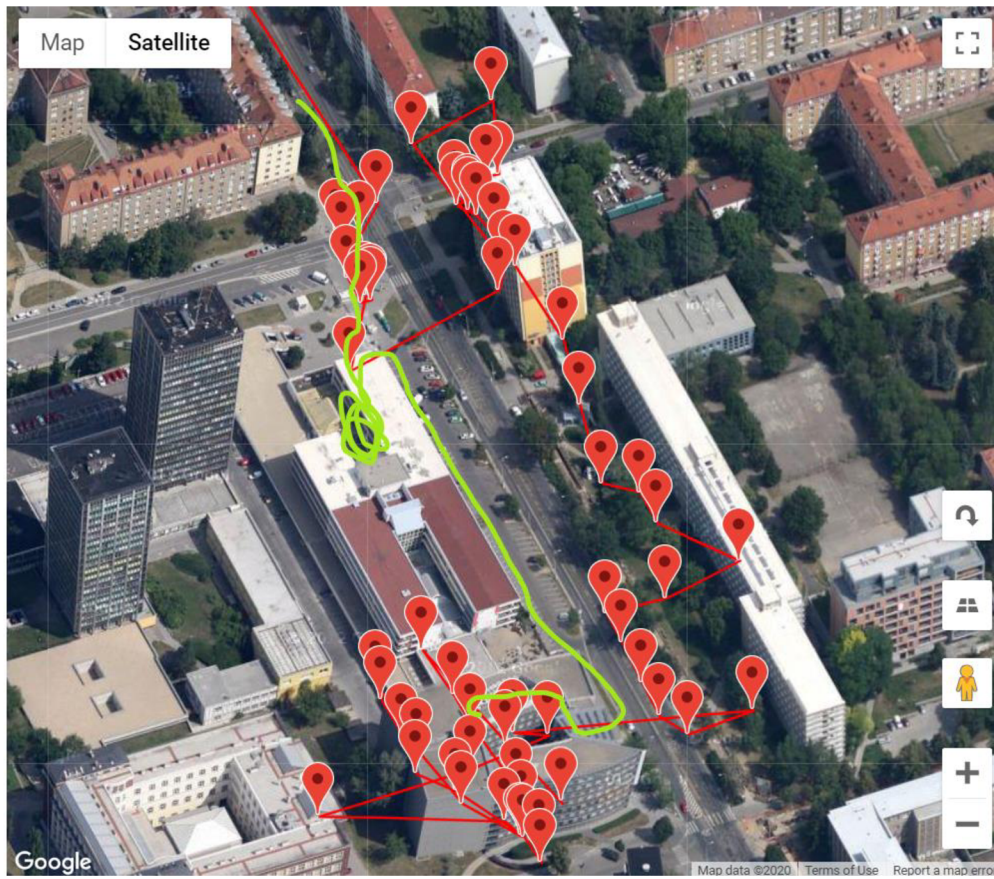
Inside buildings:

Figure 4.5 green line is the real track(imaginary line drawn based on knowing the area and the journey), red lines are the real GPS reading.

Inside building showed the worst positioning, even walking beside buildings have a bad accuracy as noted from the figure 4.5, we can see the green line is the estimated track, meanwhile the red line is the GPS sent position. Active and passive antennas showed similar result with slightly better for the active antenna.

It is worth it to notice that number of satellites is important factor in accuracy determining. during this log in figure 4.5 the GPS maintained a signal from 7 satellites, and there was low accuracy, usually in a more accurate log the number of satellites is from 8 to 10.
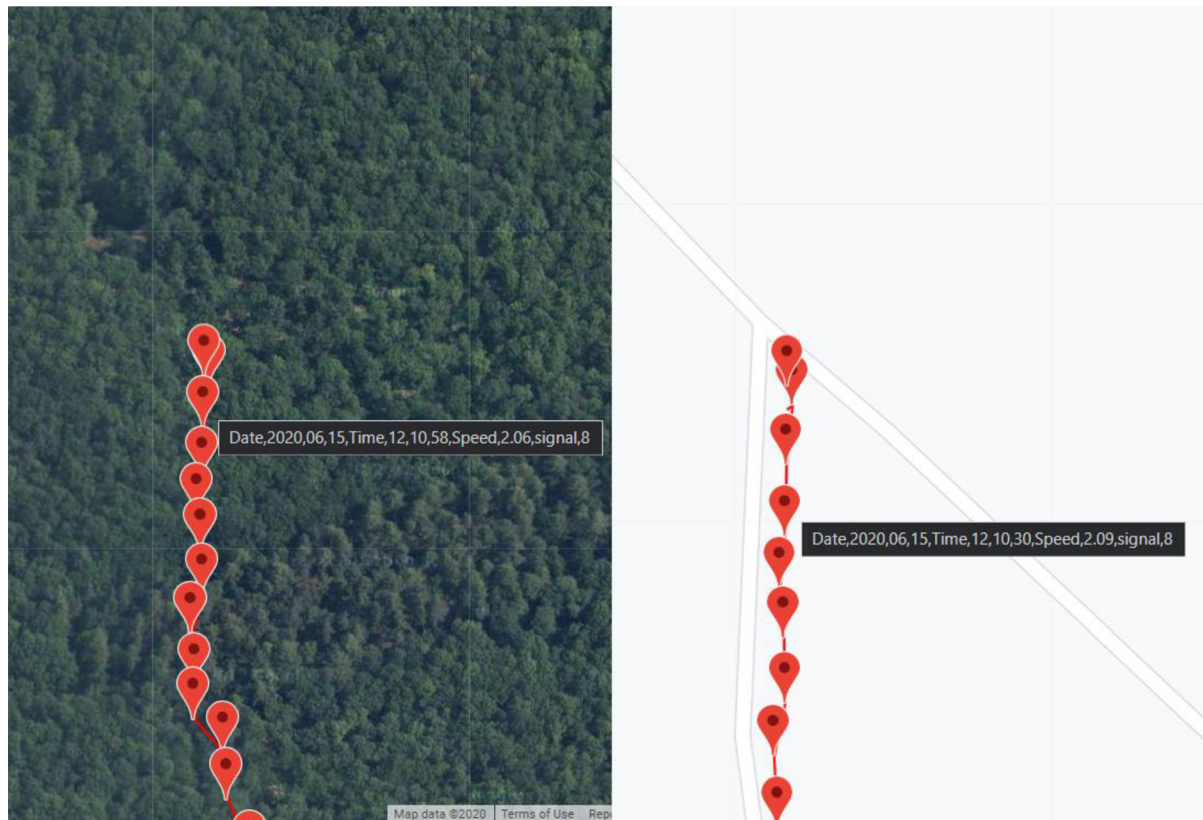
Forest and open spaces:

Figure 4.6 GPS log from hiking trip.

Both forest and open spaces showed similar and very accurate positioning. Meanwhile open spaces gave the highest number of satellites in view (up to 11 with active antenna and 10 with passive antenna).

In the figure 4.6 we can see the log on a hiking road inside the forest, to the left is the "Satellite view", and to the right is the "road view" for the same location and GPS log.

GSM signals are just like cell phones, in mountains or places with low mobile network connection, it is expected to not be able to connect and send the data.

4.2. Results and Overall real-life device functionality

By reviewing the components of the GPS tracking device both the Arduino Nano board and Sim808, and connecting them and coding, then performing tests of the device. The GPS tracker proved to be functional and reliable for sending GPS location regularly over the mobile network using the GPRS maintaining high accuracy in most cases.

The battery chosen can support the device for many hours or even days if will be used occasionally based on the tests made, giving the final device a higher level of usability. A box was made to save the components from mechanical harm via 3-D printing technology.

The Website solution showed a stable and efficient performance, with some limitations specially regarding to range being based on the log numbers rather than date or journey log. But the overall usability of the website delivers enough usability for the sake of this project.

After testing the entire system of the GPS tracker and the Web application together, the end result is fully functional, and the main objectives of the project was fulfilled, the main feature of the project that it is highly customizable and upgradable.

Upgradability and Modification:

The device can be further modified and developed, the function behind the ability to send battery capacity is the same for sensors, and the ability to respond with LED blink to server reply means a set of different replies from the server can actually extend to controlling function for the device. Moreover, this device can be made to make calls and send messages and send the GPS data in the same time.

Possible applications:

This product can be useful as a tracking device and since it is small enough and can be handled or left in the backpack it can be used for cars, pets, children monitoring.

The websites with map and saved log from some trips can be seen on the website URL:

https://vutmaizi.000webhostapp.com/A.php

References:

[1] UNITED STATES OPENING GPS DATA FOR CIVILIAN USE. Thegovlab.orgmenu [online]. [cit. 2020-06-26]. https://odimpact.org/case-united-states-opening-gps-data-for-civilian-use.html

[2] Space Segment [online]. Official U.S. government information about the Global Positioning System (GPS) and related topics [cit. 2020-06-24]. https://www.gps.gov/systems/gps/space/

[3] GPS will be accurate within one foot in some phones next year [online]. [cit. 2020-06-26]. Dostupné z: https://www.theverge.com/circuitbreaker/2017/9/25/16362296/gps-accuracy-improving-one-foot-broadcom

[4] What are various GNSS systems? [online]. [cit. 2020-06-26]. Dostupné z: https://www.geospatialworld.net/blogs/what-are-the-various-gnss-systems/

[5] Glonass: Has Russia's sat-nav system come of age? [online]. [cit. 2020-06-26]. Dostupné z: http://news.bbc.co.uk/2/hi/8595704.stm

[6] GLONASS Performances [online]. [cit. 2020-06-26]. Dostupné z: https://gssc.esa.int/navipedia/index.php/GLONASS_Performances

[7] The opening ceremony of GSA Agency [online]. [cit. 2020-06-26]. Dostupné z: https://www.czechspaceportal.cz/en/section-7/news/the-opening-ceremony-of-gsa-agencys-headquarters-was-attended-by-vips-in-space-activities--satellite-navigation.html

[8] New satellite launch extends Galileo's global reach [online]. [cit. 2020-06-26]. Dostupné z:https://www.esa.int/Applications/Navigation/New_satellite_launch_extends_Galileo_s_global_reach

[9] Galileo Performances [online]. [cit. 2020-06-26]. Dostupné z: https://gssc.esa.int/navipedia/index.php/Galileo_Performances

[10] China GPS rival Beidou starts offering navigation data [online]. [cit. 2020-06-26]. Dostupné z: https://www.bbc.com/news/technology-16337648

[11] Munich Summit Charts Progress of GPS, GLONASS, Galileo, Beidou GNSSes [online]. [cit. 2020-06-26]. Dostupné z: https://insidegnss.com/munich-summit-charts-progress-of-gps-glonass-galileo-beidou-gnsses/

[12] How GPS Receivers Work [online]. [cit. 2020-06-26]. Dostupné z: https://electronics.howstuffworks.com/gadgets/travel/gps3.htm

[13] How GPS Works [online]. [cit. 2020-06-26]. Dostupné z: https://www.maptoaster.com/maptoaster-topo-nz/articles/how-gps-works/how-gps-works.html

[14] Frequently Asked Questions [online]. Arduino [cit. 2020-06-26]. Dostupné z: https://www.arduino.cc/en/main/FAQ#toc11

[15] ARDUINO NANO [online]. [cit. 2020-06-26]. Dostupné z: https://store.arduino.cc/arduino-nano

[16] What is GNSS? [online]. [cit. 2020-06-26]. Dostupné z: https://www.oxts.com/what-is-gnss/

[17] GPS device reviews [online]. [cit. 2020-06-26]. Dostupné z:
https://wiki.openstreetmap.org/wiki/GPS_device_reviews

[18] All You Wanted to Know About GSM Module and GPRS Module [online]. [cit. 2020-06-26]. Dostupné z: https://www.electronicsforu.com/resources/gsm-module

[19] General Packet Radio Service (GPRS) [online]. [cit. 2020-06-26]. Dostupné z:
https://www.twilio.com/docs/glossary/what-general-packet-radio-service-gprs

[20] All You Wanted To Know About AT And GSM AT Commands [online]. [cit. 2020-06-26]. Dostupné z: https://www.electronicsforu.com/resources/cool-stuff-misc/gsm-at-commands

[21] NEO-7 series [online]. [cit. 2020-06-26]. Dostupné z: https://www.u-blox.com/en/product/neo-7-series

[22] SIM808 GSM / GPRS + GNSS [online]. [cit. 2020-06-26]. Dostupné z:
https://simcom.ee/modules/gsm-gprs-gnss/sim808/

[23] SIM808_Hardware Design_V1.02 [online]. In: . [cit. 2020-06-26]. Dostupné z:
http://www.elecrow.com/wiki/images/e/e8/SIM808_Hardware_Design_V1.02.pdf

[24] AnalogRead() [online]. [cit. 2020-06-26]. Dostupné z:
https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

[25] FT232R USB UART IC Datasheet [online]. In: . [cit. 2020-06-26]. Dostupné z:
https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[26] A Guide to Understanding Battery Specifications. MIT Electric Vehicle Team,
December 2008. [cit. 2020-06-26]. Dostupné z:
http://web.mit.edu/evt/summary_battery_specifications.pdf

[27] The Ultimate Filament Guide [online]. [cit. 2020-06-26]. Dostupné z:
https://all3dp.com/1/3d-printer-filament-types-3d-printing-3d-filament/

[28] SIM800 Series_GNSS_Application Note_V1.00 [online]. In: . Shanghai SIMCom
Wireless Solutions, 2015-04-10 [cit. 2020-06-26]. Dostupné z:
http://www.elecrow.com/wiki/images/0/05/SIM800_Series_GNSS_Application_Note_V1.00.pdf

[29] Arduino GPS tracker, SIM808 version [online]. [cit. 2020-06-26]. Dostupné z:
https://www.iforce2d.net/sketches/

[30] SIM800 Series_AT Command Manual_V1.09 [online]. In: . Shanghai SIMCom
Wireless Solutions, 2015-08-03 [cit. 2020-06-26]. Dostupné z:
http://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf

[31] Simple Polylines [online]. [cit. 2020-06-26]. Dostupné z:
https://developers.google.com/maps/documentation/javascript/examples/polyline-simple

[32] Better use of Global Satellite Systems for Safer and Greener Transport - Scientific
Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Position-computation-by-trilateration_fig4_305724028 [accessed 7 Jun, 2020]

[33] WHAT IS DGPS DIFFERENTIAL GPS? HOW DOES IT WORK? [online]. In: . [cit. 2020-06-26]. Dostupné z: https://studymaterialonline.com/what-is-dgps-differential-gps-how-does-it-work-dgps-pdf/

[34] Neo-7M Compatible GPS Module [online]. [cit. 2020-06-26]. Dostupné z: https://www.minikits.com.au/neo-7m

[35] GSM Shield pro Arduino, SIM900 [online]. [cit. 2020-06-26]. Dostupné z: https://www.gme.cz/gsm-shield-pro-arduino-sim900

[36] Crowtail- SIM808 [online]. [cit. 2020-06-26]. Dostupné z: https://www.elecrow.com/wiki/index.php?spm=a2g0o.detail.1000023.17.24c871e9b4LUVx &title=Crowtail-_SIM808

[37] Discharge Characteristics of Li-ion [online]. [cit. 2020-06-26]. Dostupné z: https://batteryuniversity.com/learn/article/discharge_characteristics_li

List of attachment:

these are the attachments for codes used in this project. There are 2 folders:

arduino_code_sim808 : the Arduino code.

Website : the files needed for the website.

README : text file for reviewing each folder contents.