

Real Time Vehicle Tracking System and Energy Reduction

by

Salman Almishari

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

© Salman Almishari 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Technology has been growing rapidly and various fields of technology can be combined to advance societies. The Internet of Things (IoT) is one of these technologies that combines and connects a variety of things to come up with some more beneficial information. In this thesis, an example of an IoT portable system has been built and tested.

The system is called Smart Vehicle System (SVS), and includes three main parts: the Tracking Unit, Cloud, and Android application. The Tracking Unit is positioned inside a vehicle to sense the vehicle's temperature, speed, and location, then uploads them to the cloud via GSM network. All the components and communication of this Tracking Unit are described in detail. The cloud includes a web panel to view and manage the data, web server to do data processing, and database to store all the information. The Android application is used to receive notifications and to view the vehicle's current temperature and location.

Some constraints are included in the system to notify the administrator and the driver of certain events. The emphasis is on two main constraints: high or low temperatures inside the vehicle and location restrictions. The administrator sets up these constraints using the web panel. If there is any violation of these constraints, notifications are issued and sent to the administrator via email and to the Android application. The processes of these constraints are described in detail. This system is intended to help transportation companies manage their fleets more effectively.

The SVS is a portable system, and so it functions on batteries. Therefore, a power reduction algorithm is recommended and examined. We have performed 19 different experiments, before and after applying the proposed algorithm; four of them are dynamic and 15 are at a fixed location. With the power reduction algorithm, we are able to reduce the energy consumption of the tracking unit up to 17%. All the setup and results are presented. A Monsoon Power Monitor device and a laptop are used to measure the power consumption and compare the results. We conclude that there is a need for data processing before uploading results to the cloud, which saves power and database size.

Acknowledgment

I want to thank Prof. Sagar Naik for guiding my study on this thesis with his patience, motivation, enthusiasm, and immense knowledge. His supervision assisted in discovering some hidden points of this research and in writing this paper. He is one of the best advisers that I have ever met during my study.

Dedication

This is dedicated to my father, mother, and wife who have supported me in completing this research and all of the ones I love. Likewise, special thanks to Professor Naik for his guidance and support. A special mention goes to my daughter Algohara who bring me joy every day, and my son Nawaf who was not born until the end of this efforts to succeed.

Thanks also to Mary McPherson of UW's writing and communication center for her thoughtful comments on my writing.

Table of Contents

List of Tables	viii
List of Figures.....	ix
List of Abbreviations	xi
1 Introduction.....	1
1.1 Motivation.....	3
1.2 Problem Statement.....	4
1.3 Solution Strategy and Contributions.....	5
1.4 Organization of The Thesis	7
2 Background.....	8
2.1 Global Positioning Systems (GPSs)	8
2.2 Cloud Computing.....	9
3 Literature Review.....	12
3.1 Vehicle Tracking System Technologies	12
3.2 Modules for Power Consumption Monitoring.....	15
4 System Design.....	19
4.1 Tracking Unit (Sensors and Middleware)	23
4.1.1 Microcontroller (Arduino UNO R3)	23
4.1.2 Temperature Sensor (DS18B20).....	25
4.1.3 GPRS/GPS Quadband Module (SIM908)	27
4.1.4 GPS Antenna.....	30
4.1.5 GSM/GPRS Antenna	32
4.1.6 9V Battery and Barrel Jack Adapter	32
4.1.7 Tracking Unit Communication.....	33
4.2 Cloud.....	39
4.2.1 Web Host and Domain Name.....	39
4.2.2 Website Design	39
4.3 Android Application.....	47
4.3.1 Login Page.....	47
4.3.2 Home Page	48
4.3.3 Profile Page.....	49
4.3.4 About Page.....	50
5 Solution, Experiments and Results	52
5.1 Suggested Power Saving Methodology for the Tracking Unit	52
5.2 Experimental Setup.....	55
5.3 Experimental Scenarios.....	59
5.4 Experiment for One Hour Static Location	61
5.4.1 Detailed Results for each experiment	62
5.5 Experiment for One Hour Static Location Without Temperature Sensor...	66
5.6 Dynamic Experiment for One Hour	71
5.6.1 Real-Live Dynamic Experiment Scenario	73
6 Challenges, Conclusion, and Future Work	78
6.1 Challenges.....	78
6.2 Conclusion	81

6.3	Future Work.....	82
7	References.....	83
8	Appendix.....	86

List of Tables

Table 1: Jumpers Arrangement for GPRS/GPS to be Power from Arduino Board	28
Table 2: Fixed Experiment for 10 Seconds Results	62
Table 3: Fixed Experiment for 20 Seconds Results	63
Table 4: Fixed Experiment for 30 Seconds Results	63
Table 5: Fixed Experiment for 40 Seconds Results	64
Table 6: Fixed Experiment for 50 Seconds Results	64
Table 7: Average Power Comparison Between Normal and Power Saving Mode.....	66
Table 8: Comparison of TU Measurements for 10 Seconds Upload Frequency Results	67
Table 9: Comparison of TU Measurements for 20 Seconds Upload Frequency Results	68
Table 10: Comparison of TU Measurements for 30 Seconds Upload Frequency Results	68
Table 11: Comparison of TU Measurements for 40 Seconds Upload Frequency Results	69
Table 12: Comparison of TU Measurements for 40 Seconds Upload Frequency Results	70
Table 13: Dynamic Experiment for 10 Seconds Results	74
Table 14: Dynamic Experiment for 40 Seconds Results	76

List of Figures

Figure 1-1: Number of Vehicles in the World [2]	1
Figure 1-2: Block Diagram of Smart Vehicle System.....	3
Figure 3-1: End Device Battery Lifetime for Different Packet Size [1].....	17
Figure 4-1: System Overview (Software Module).....	20
Figure 4-2: System Hardware Module.....	22
Figure 4-3: Tracking Unit Components.....	23
Figure 4-4: Arduino Board	24
Figure 4-5: Temperature Sensor.....	25
Figure 4-6: Arduino and Temperature Sensor Physical Connection	26
Figure 4-7: Physical Connection Between Arduino and GPS/GPRS Module.....	28
Figure 4-8: GPRS/GPS Shield Diagram (Face Up):	29
Figure 4-9: GPRS/GPS Shield Diagram (Face Down).....	30
Figure 4-10: GPS Antenna.....	31
Figure 4-11: Connection of GPS and GSM/GPRS Antennas with GPS/GPRS Module	31
Figure 4-12: GSM Antenna	32
Figure 4-13: 9V Battery and its connector	33
Figure 4-14: Tracking Unit System Code in Phase 1	34
Figure 4-15: Tracking Unit System Code in Phase 2	35
Figure 4-16: Tracking Unit System Code in Phase 3	36
Figure 4-17: Tracking Unit Communication Flow Chart	38
Figure 4-18: Login Page.....	40
Figure 4-19: Home Page.....	41
Figure 4-20: Details Page.....	42
Figure 4-21: Edit Details Page.....	42
Figure 4-22: Edit User Information Page.....	43
Figure 4-23: History Page.....	44
Figure 4-24: Temperature Page	45
Figure 4-25: Edit Temperature Range Page	46
Figure 4-26: Event Page	46
Figure 4-27: Android Application Login Page	47
Figure 4-28: Android Application Home Page.....	48
Figure 4-29: Android Application Profile Page	49
Figure 4-30: Android Application About Page	50
Figure 5-1: Sequence Message for Power Saving Algorithm with Different Stages...	54
Figure 5-2: Monsoon Application GUI.....	57
Figure 5-3: Power Connection Between Monsoon Power Monitor and Arduino Board	58
Figure 5-4: Highest and Lowest Temperature for Vehicles.....	60
Figure 5-5: University of Waterloo as a Radius Location.....	61
Figure 5-6: Power Saving Percentage for all of the 10 Experiment	65
Figure 5-7: Average Power Comparison Between Normal and Power Saving Mode for Tracking Unit	66

Figure 5-8: Comparison of Average TU Power, with and without Temperature Sensor	71
Figure 5-9: Planed Dynamic Experiment Route	72
Figure 5-10: Real Live Dynamic Experiment Route	73
Figure 5-11: Power Measurement for 10 Seconds with Normal Mode	75
Figure 5-12: Power Measurement for 10 Seconds with Power Saving Mode	75
Figure 5-13: Power Measurement for 40 Seconds with Normal Mode	76
Figure 5-14: Power Measurement for 40 Seconds with Power Saving Mode	77
Figure 5-15: Comparison Between Normal Mode and Power Saving Mode	78
Figure 6-1: SIM 908 (GPS & GSM) Short Antennas	79

List of Abbreviations

6LoWPAN - IPv6 over Low Power Wireless Personal Area
ANSI - American National Standards Institute
APN: Access Point Name
ARM - Advanced RISC Machines
AT – Attention
AVL - Automatic Vehicle Location
CoAP - Constrained Application Protocol
DNS - Domain Name System
GPRS - General Packet Radio Service
GPS - Global Positioning System
GSM - Global System for Mobile communication
GUI - Graphical User Interface
HTTP - Hypertext Transfer Protocol
ICSP - In-Circuit Serial Programming
IDE - Integrated Development Environment
IEC - International Electrotechnical Commission
IEEE - Institute of Electrical and Electronics Engineers
IETF - Internet Engineering Task Force
IOT - Internet of Thing
IP address - Internet Protocol address
JSON - JavaScript Object Notation
LCD - Liquid Crystal Display
LED – Light Emitting Diode
mA - milliAmpere
MAC - Media Access Control
mAh - milliAmpere per hour
MPM - Monsoon Power Monitor
ms - millisecond
mW - milliWatt
MySQL – My Structured Query Language
NMEA - National Marine Electronics Association
PC - Personal Computer
PHP - Hypertext Preprocessor
PIN - Personal Identification Number
SIM - Subscriber Identity Module
SMS - Short Message Service
STCA - Spatio-Temporal Correlation Algorithm
SVS - Smart Vehicle System
TCP/IP - Transmission Control Protocol/Internet Protocol
TU - Tracker Unit
UART - Universal Asynchronous Receiver-Transmitter
URL - Uniform Resource Locator
USB - Universal Serial Bus
WWW - World Wide Web

Chapter 1

1 Introduction

Technology is growing rapidly. Fields of technology can be combined to improve lives and comfort levels. One very popular growing technology is the Internet of Things (IoT), which plays a major role in connecting objects through network connections. As stated by [1], the number of connected devices in 2020 will exceed the number of people worldwide by more than seven times, and Cisco anticipates that in 2020 the number of connected devices will reach 50 billion. Communications helps to build intelligent objects that make decisions based on received information. IoT helps big companies to improve efficiency and reduce overload work. For example, one of the most important aspects of managing a fleet of vehicles is to get each vehicle's location, speed, and temperature. According to the Statistics Portal website, there were more than 1.2 billion vehicles in the world in 2014 (Figure 1-1) [2]. Managing these vehicles can be challenging, but with the new technologies, it can be done. Many companies keep track of their vehicles with an integrated an Automatic Vehicle Location (AVL). Such devices can also be used to improve transportation efficiency and safety by analyzing data for future strategies.

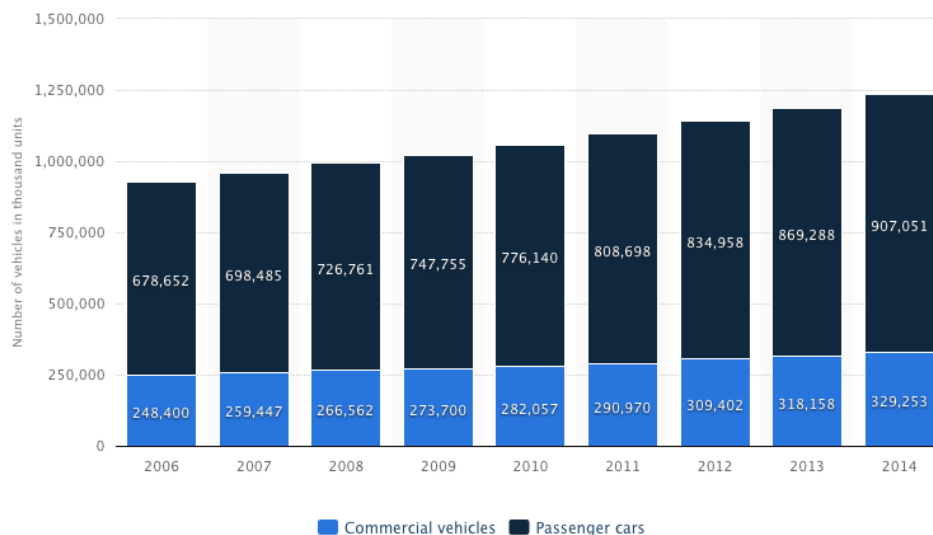


Figure 1-1: Number of Vehicles in the World [2]

In this research, a portable Smart Vehicle System (SVS) has been built. This system notifies a user of special events as they happen in the vehicle, such as moving out of a specific range of location or temperature. SVS has three main components: a Tracking Unit (sensors and middleware), cloud, and smartphone. The sensors will sense and send data to the middleware. The middleware stores, filters, and sends the data to the cloud. Then, the cloud stores and analyzes the data in a web panel. This web panel gives the administrator the ability to manage data and users, and to set up constraints as needed. If there is a special event after analyzing the data on the cloud, a notification will be sent to the smartphone and an email will be sent to the administrator of the system. The smartphone is used to receive notifications and to view the data that has been collected.

The block diagram of the Smart Vehicle System on Figure 1-2 shows how the entire system actually works. The tracking unit should be placed inside the vehicle that is to be tracked. This unit has a GPS receiver, which collects coordinates and determines the speed of the vehicle from the satellite. In addition, it senses the temperature inside the vehicle. Then, it sends the collected data to the Global System for Mobile communication (GSM) tower by means of the GSM modem. The data is then sent to the cloud via the Internet, where it is stored in the database and used to indicate the vehicle's location on Google Maps. The user can also see the location of the vehicle on a mobile phone or a web application. Any notification issued is sent to the user's Android application and emailed to the web application administrator.

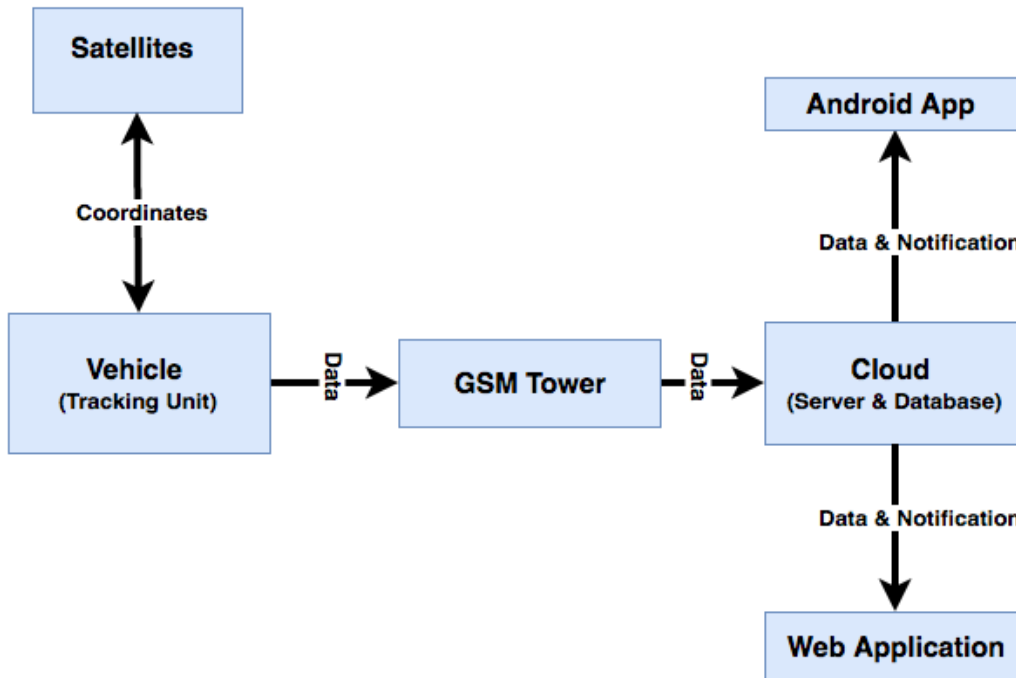


Figure 1-2: Block Diagram of Smart Vehicle System

One of the main focuses of this research is the sensors and middleware in the Tracking Unit that are placed inside the vehicle. The Unit consumes a lot of power, which will be studied in detail later in the thesis. Hence, a power reduction algorithm has been introduced and applied to the developed system.

1.1 Motivation

The number of IoT applications continues to grow. One of the most popular and ever-growing applications is portable tracking systems that work on batteries. Currently, self-driving cars are an exciting innovation and one of the main research trends. Self-driving cars are guided by connected sensors that report to the main computer to further process the data and make valuable information from it. The research of this thesis was motivated by a desire to build a fully portable functional tracking system that has the ability to track location, speed, and temperature of a vehicle and then improve its performance. These functionalities use sensors to sense the data and report it to other devices for further processing. After building the system, an improvement of the system was planned:

reducing its energy consumption to make it as functional as possible. Hence, learning how the system works and trying to build it are the keys to this work.

Battery technologies have not been improved significantly in the last 25 years, but IoT applications are consuming more and more power [3]. The lifetime of any portable IoT application will depend on its battery life; hence, there is a need to optimize the battery life of IoT applications.

Furthermore, transportation companies are moving to technological systems to help them manage their fleets of vehicles. Such systems will help these companies locate their vehicles and learn if the vehicle passes a certain limit. This ability will reduce theft and breaches of contract. In addition, because excessive temperatures inside a vehicle can cause fires or damage properties, this system will alert the company and the user if certain temperature values are reached.

1.2 Problem Statement

The number of vehicles in the world is increasing, and by the end of 2035 it will reach 2 billion, according to green car reports [4]. There is a need for applications that track vehicles locations and temperatures so as to easily managing fleets, improve efficiency, and prevent hazards. The ability to manage fleets of vehicles is critical but still needs applications that enable it. Transportation companies are currently losing enormous amounts of money because of weather effects or inexperienced employees. Tracking vehicle behavior gives firms the ability to organize their transport in an efficient way and provide higher standards, which could save time and money and prevent risks. The vehicle tracking applications could be portable, or permanently installed in vehicles. Our focus is on the portable devices.

One of the problems that most companies face is that employees using company vehicles do not always follow the company guidelines: for instant, they may use the vehicle off limits in certain areas, go faster than the speed limit, or not pay attention to the temperature of the cargo or inside the vehicle. The goal of this work is to build a

portable system, called Smart Vehicle System (SVS), that uploads vehicular data to the cloud and then make some decisions according to certain constraints.

We have consequently developed the SVS system which includes embedded devices for uploading data to the cloud. The design of applications running on these embedded devices has an impact on the devices' power consumption. Hence, the more sophisticated the design, the less the battery life. In addition, the process of fetching the data from the sensors and storing it in these embedded devices consumes significant power. The problem is that battery life at present is very limited, even though many researchers are trying to improve it. However, these improvements are occurring very slowly compared to the development of new device functionalities. Therefore, there is a need to improve the software side to accommodate the growth of applications such as SVS. More research is needed to reduce the power consumption from the software side and so help improve battery life and prolong device live.

1.3 Solution Strategy and Contributions

Our solution for managing a fleet of vehicles is to build a portable system that tracks and uploads the location, speed, and temperature of a vehicle to the cloud for further processing. The power consumption of such a portable tracking system is critical. In this work, a study of the power consumption is introduced and a power reduction algorithm is applied to reduce the power consumption of the embedded devices. Developers must consider the amount of data that should be sent to the cloud. In addition, the process of fetching data from the sensors and storing it in these embedded devices consumes a large amount of power. The following paragraphs describe a detailed solution strategy with four main components.

First, a portable Tracking Unit is built and contains embedded devices such as the Arduino, GPRS/GPS Quadband Module, and some sensors, and its functions are to observe the surrounding environment and upload its readings to the cloud. The Arduino board is the brain of this Tracking Unit, where all the developed code is running. The software and the hardware components are described in detail in Chapter 4.

Second, a cloud system is developed using a cloud service, where all the readings that are uploaded from the tracking system are stored and analyzed. When the readings are received, a check of all necessary variables is performed. Then, the readings are stored in the database and examined to determine whether any constraints have been violated. If there is a violation, a warning is sent to the administrator and the user.

Third, an Android application is developed to receive the warnings, to locate the vehicle, and to report real-time temperatures. This application contains a temperature bar that shows the maximum and the minimum temperatures set by the administrator. It also displays the current temperature of the vehicle. Using embedded Google maps, an exact location of the vehicle is shown in a map in relation to city, province and country.

Finally, a power consumption analysis is presented and a power reduction algorithm is developed. The power-reduction algorithm includes adding two stages, the store and compare stages, to the process, via the Arduino code before uploading the data to the cloud. The store stage stores the data in the Arduino's memory, and the compare stage compares previously stored data sent by the sensors with the current data. We have conducted ten static and four dynamic experiments, before and after applying the algorithm, which demonstrated that the algorithm improves the average power consumption of the Tracking Unit by up to 17.6%. We used various upload frequencies to apprehend the difference in power saving. For instance, when we used a 10-second sampling frequency in a fixed location experiment, the power saving mode, which includes our algorithm, consumed only 631mW power on average, whereas the normal mode consumed 766mW. All of the implementation details are described in the experiments section in Chapter 5.

In summary, this research provides the following contributions

- We developed a portable Tracking Unit system that can be placed in a vehicle
- We developed a cloud-based system for storing and managing the data provided by the physical tracking system
- We developed an Android application to receive notifications and show the status of the vehicle.

- We introduced an algorithm to reduce the power consumption of the developed Tracking Unit system.
- We presented experimental setups and results that demonstrate that our algorithm's reduction of Tracking Unit power consumption .

1.4 Organization of The Thesis

The rest of the thesis is organized as follows. Chapter 2 briefly presents the background of the GPS systems and cloud services. Chapter 3 reviews the literature related to tracking systems and energy reduction in embedded systems. Chapter 4 presents an overview of the system design and its software and hardware components, and also we discuss the specifications for each component of the entire system and how they communicate with each other. Chapter 5 presents the algorithm suggested for reducing the power consumption of the developed tracking system and discusses the experiment setup, scenarios, and results. Finally, Chapter 6 discusses some of the challenges in developing and testing the system, then concludes the thesis.

Chapter 2

2 Background

Vehicle tracking systems are a means of monitoring vehicles, detecting certain activities, then notifying users. Such systems use various kinds of technologies, which are covered under the umbrella of IoT. One of the most important technologies used for these systems is the Global Positioning System (GPS), which uses satellites to determine the exact location of an object.

2.1 Global Positioning Systems (GPSs)

Recently, the GPS has become one of the most used, efficient, and precise technologies, and is used to locate objects, persons, or assets for outdoors or in an open atmosphere [5]. A GPS involves connecting to satellites that fly in the orbit of the earth, and is thus considered a space-based navigation system. A GPS involves into three sections [6]. The first is the space section, which includes at least 24 satellites orbiting the earth twice a day. The second, the control section, includes five different stations on the ground, which monitor and manage the satellites by updating the orbit data and clock correction. The third is the user section, which consists of all different kinds of GPS receivers that are able to receive data from the satellites. After a GPS receiver has located its position from the satellites, the information is usually presented in one of two formats: the latitude and longitude or Universal Transverse Mercator (UTM).

In the system devised for this research, the GPS receiver provides information about the location and time of any object anywhere and anytime as long as there is an unobstructed line of sight to at least three or more GPS satellites [6]. These satellites were originally used for military applications and controlled by the U.S. Department of Defense and NAVigation Satellite Timing and Ranging [6]. In general, the satellites frequently send signals to GPS receivers. After the GPS receiver obtains a signal from at least three satellites, a triangulation technique is used to calculate the receiver's two-

dimensional position; that is, its latitude and longitude [7]. If a GPS receiver obtains a signal from four or more satellites, it is able to calculate its three-dimensional position: its latitude, longitude, and altitude. After the location id is obtained, the receiver calculates the vehicle's average speed and movement direction.

The drawbacks of using GPS technology are that it [5] [7]:

- Consumes substantial amounts of power.
- Cannot detect locations indoors.
- Needs an unobstructed line of sight to at least three satellites.
- Provides only approximated locations, within 10 to 15 feet.
- Performs poorly in overcrowded areas.

2.2 Cloud Computing

Cloud Computing is a general term for any hosted services provided through the Internet. The theory of cloud computing was introduced in 2006 and is revolutionizing the next generation of the Internet [8]. The simple cloud shapes used in diagrams and flowcharts were the inspiration behind Cloud Computing [9]. Today, there are three different main categories of cloud computing hosted services: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [8] [9] [10]. These categories are described in more detail next:

1. **Infrastructure-as-a-Service (IaaS):** This kind of service provides virtual computers or servers, space for storing data, Application Program Interfaces (APIs) to give users the ability to manage their virtual machines, storage, protective firewalls, load balancing, and other network computing resources [8]. Some of the most popular providers are Amazon Web Services (AWS), Google Compute Engine (GCE), and Microsoft Azure.
2. **Platform-as-a-service (PaaS):** PaaS contains many diverse software development tools hosted on the provider's infrastructure. This kind of service makes the development, testing, improvement, and implementation of software

much easier, quicker, and more cost effective [11]. PaaS benefits organizations as they can reduce the aggregate of coding needed. Apprenda is one of the leading providers that offer PaaS for Java and .NET.

3. **Software-as-a-service:** the provider offers clients access to software running on their clouds, and these clients can use it either through a web browser or an application. SaaS is the most used service in the market, and has been used in developing our Smart Vehicle System.

Furthermore, three distinct types of clouds can be used [10] [12]:

1. **Public clouds:** Public clouds are available to the public through service providers who host infrastructure such as data center software and hardware. Commonly, public clouds are pay-as-you-go type, and anyone can subscribe [12]. This is the kind of cloud used in developing the Smart Vehicle System. Sun Cloud and IBM's BlueCloud, Hostgator are examples that provide public clouds.
2. **Private clouds:** As the name indicates, a private cloud is a computing infrastructure for a specific organization. This kind cloud might be owned, administered, and maintained by a third party or the same organization, or a combination of both [33]. The physical location of this cloud can be on the organization's premises or a different location, and an example of a private cloud is Amazon EC2 and Microsoft Azure.
3. **Community Clouds:** Community Clouds are cloud shared for particular use by a certain community of customers with the same concerns, such as data security. This kind cloud might be owned, administered, and maintained by a third party or the same organization, or a combination of both [13]. Think of community clouds as public clouds but for specific companies or users.
4. **Hybrid clouds:** Hybrid clouds are composed of two or more clouds (private, community or public) are one of the type most used by organizations because of

their flexibility and data distribution options [13]. This kind of cloud permits workloads to be exchanged between different kinds of clouds to meet an organization's computing and cost requirements. For instance, an organization can maintain its sensitive or critical data and workload on a private cloud on its premises, and use a public clouds provider for insensitive resources.

Chapter 3

3 Literature Review

This chapter provides a brief overview of the theories and technologies relevant to the proposed system.

3.1 Vehicle Tracking System Technologies

Researchers have developed systems similar to our SVS system but with different designs and hardware, or for other applications such as health. Such systems can be developed in many different ways. Ours focuses on the real-time sensing of location, speed, and temperature of a vehicle and then adds features by setting constraints for these variables. Examples of other systems are in [14], [15], [16] and [17], but these focus on vehicle location and how to present it on Google. On the other hand, systems like [15], [18], and [19] use Short Message Service (SMS) to ask the unit installed inside the vehicle location. Our system, in contrast, is more precise, in that it sends the location and other information continuously, at specified times, to the cloud.

For decades, preventing vehicle theft has been a main research topic, and many researchers such as [17] [19] have installed extra vehicle parts to prevent a vehicle from moving in case of theft. Systems similar to [19] have the ability to receive an SMS command from the vehicle's owner, to turn off the engine. Cost has also been a major interest of researchers, for instance [14], who employs open resources with low hardware cost. Most researchers use the vehicle as a power source for their system while [17] and our system use only its own battery. The following paragraphs introduce similar research and describe each study in more detail.

The authors of reference [14] state that recent vehicles are equipped with innovative technologies developed by the vehicle manufacturers for their own use and not available to the public. Most of these systems have been established using copyrighted software and hardware. Due to these barriers, the authors of [14] designed and developed

a system using open source hardware and software with low cost. The system was developed for a mobile real time GPS tracker that includes Google Maps API V3. The system contains two main components: embedded devices and sensors inside the vehicle (vehicle unit) and a web server that stores and retrieves information, then shows it on an embedded Google Map. The location of the vehicle is uploaded every second from the vehicle unit to the web server, which makes it a real-time system. The vehicle unit transfers the data using Transmission Control Protocol/Internet Protocol (TCP/IP) via a GPRS connection to a specified web server. Then, the server stores the information in an explicit database, to be retrieved when a user's browser acquires this information. A JavaScript code is developed to incorporate this information to the Google Maps API that presents the vehicle's location on a map. Further details of the system's design and testing are available in the paper.

Kotte and Yanamadala [21] have developed a GPS tracking/navigation system that obtains a vehicle's information using a GPS and GSM network, and then sends the location information of a vehicle to a tracking server for viewing on Google Earth. This information is sent at specified time intervals, set by the user on a web application. Their system model consists of two major components: the In-Vehicle unit and a Tracking server/monitoring station. The In-Vehicle unit contains a microchip (PIC18F248), GM862-GPS GSM/GPRS modem, Liquid Crystal Display (LCD), GPS antenna, and GSM antenna. The tracking server has software for viewing the data and storing information in the database. To acquire information from the In-Vehicle unit, this system uses Short Message Service (SMS) initiated from the tracking server at intervals specified by the user on an application. Then the In-Vehicle unit sends the vehicle's information via GPRS back to the tracking server. The key objective of this work was to learn how build the system.

The system proposed in [16] is designed to help with public transportation such as taxis and buses, by means of a unit installed in these vehicles to receive SMS, from a cell phone. The system uses GPS and GSM to obtain a vehicle's location from satellites and sends this information as a text via SMS. The system model consists of an Advanced RISC Machines (ARM) processor, GSM modem with antenna, and GPS receiver with

antenna. The designer used UART0 at the 9600-baud rates and UART1 at the 4800-baud rates for communication between the processor and the GSM modem. The objective of this system was to control an LED light that is connected to the processor and obtain the location of the object. If the processor receives a text message that contains 0, 1, or 2, then accordingly it either turns the LED OFF, turns the LED ON, or acquires the location and sends it back. This work focuses only on SMS, and the data is not stored for further processing. After the location of the vehicle is sent, the data is deleted.

In [16], the authors have designed and implemented a GPS/GSM tracking system that works on batteries. This system consists of two main components: the transmitting unit and the monitoring unit. The transmitting unit, which is placed inside a vehicle that needs to be tracked, includes six distinct parts: an AT mega microprocessor, GPS module, GSM module, MAX 232 to link the AT mega with the GPS and GSM modules, a 16x2 LCD, and a 9V battery as a power source for the unit. The monitoring unit has only two parts: a GSM mobile to acquire the location from the transmitting unit and a web application to display the location. The main objective of this work is to create a system that prevents theft, using GPS/GSM technologies.

In [19], the authors have focused on security and theft prevention for two-wheeled vehicles. They developed a tracking system based on GSM/GPS communication using smartphones. The entire system consists of an Atmega162 microcontroller as an interface for the GPS and GSM modules, a 16x4 LCD display to show messages to the user, a solenoid valve, a flow control valve, a relay driver circuit, and a Smartphone for SMS communication. The design applied on this system gives the owner of the two-wheeler the ability to track its location, speed, and other parameters. The system uses a battery as a backup if the two-wheeler does not provide power. The owner of the two-wheeler sends SMS to the unit installed on the vehicle and the unit responds. Another method to get information from the unit installed is a web gateway or an Android application. The authors claim that their system outperforms current systems in reliability, control, service, and cost.

After going through these papers, we conclude that most researchers have developed systems for several reasons and using distinct designs. For example, in [18] the hardware, setup, and connection differ from those of our system. In [17], [18], and [19] the system tries to obtain vehicle location information through requests by SMS or the server. They also use LCDs to allow drivers to view their vehicle's information while driving, and most power their on-board monitoring devices from the vehicle itself. In contrast, our system requires no requests as information is uploaded automatically, and it operates on battery. Although [17] powered their system using a 9V battery, they did not explore the amount of power consumed or the lifespan of such batteries. We, however, have investigated and optimized both.

3.2 Modules for Power Consumption Monitoring

Power consumption of remote sensors is a critical dilemma for IoT systems, especially ones that run on batteries or use energy-harvesting techniques [20] [21]. Many researchers (e.g., [1], [21], and [22]) have been studying Wireless Sensor Network (WSN) to reduce the power consumption of devices or sensors connected to a network. The existing protocols for communication in WSN are not built to be power aware and further research is needed [21]. Electrical and Electronics Engineers (IEEE) standards and the Internet Engineering Task Force (IETF) expended major efforts in initiating new stacks of protocols to cover the gap in power aware communication protocols [21]. Occasionally, communication protocols contain information that does not apply for WSN, and because of that many scientists are trying to eliminate this kind of information [20]. This sub-sectional reviews studies that have been conducted to accommodate the need for power conservation.

The authors of [21] introduced a wireless communication stack that the industry considers to be power efficient, reliable, and that communicates through the Internet. The authors stress the importance of power consumption of IoT devices, particularly ones powered by batteries. They claim that existing Internet protocol stacks are not designed for such devices and consume too much power. Examples include the Hypertext Transfer Protocol (HTTP) and Transmission Control Protocol (TCP), which are used for

communication and expend excessive energy due to the amount of header data attached to the package sent and the number package acknowledgments at the upper layers.

This paper [21] also integrates standards from the IEEE and IETF, who have introduced a stack of protocols to make IoT devices consume less power, in particular IEEE 802.15.4 Physical layer and IEEE 802.15.4e Media Access Control (MAC) layer. The communication stack that this paper presented goes through four different layers: Physical, MAC, Network, and Application, which were verified to be power efficient. After deployment and testing, the authors found that IEEE 802.15.4-2006 for the Physical layer and IEEE 802.15.4e for the MAC layer are sufficient from the power consumption point of view. For the Network layer, 6LoWPAN protocol and IETF ROLL are significant for universal connection between IoT devices, while IETF Constrained Application Protocol (CoAP) clarifies that there is no need to redevelop applications to be compatible with low-power networks. Overall, power saving can be achieved from the software side by enabling different protocols and tweaking them to meet some constraints.

Spatio-Temporal Correlation Algorithm (STCA) was developed by Shabna, Jamshid, and Kumar [20] to reduce power consumption for WSNs. The emphasis of this algorithm is to eliminate unnecessary data transmission that increases power consumption. The STCA can be deployed on both event-driven and ongoing sensor networks to expand the lifespan of a sensor node. This algorithm uses a localized approach, by instructing the temporal correlation in each node. It also utilizes the location relationship with the parent node to transmit only the headers needed. The proposed methodology aims to reduce the amount of data bytes transmitted by exponentially reducing the sampling rate. Each time a sample is recorder, a correlation examination is performed, and any redundant data is removed and not transmitted. Hence, reducing the number of data transmissions decreases the power consumption of the transmitted node. An experiment was conducted using MSP430 Launch Pad, Personal Computer (PC), Energia IDE, XBee pro module, and LM35 temperature sensors. The developed STCA succeeded in decreasing in the quantity of transmissions by up to 40%. The paper also proves that reducing the number of transmitting bytes reduces the node power consumption.

However, Villas, Boukerche, De Oliveira, De Araujo, and Loureiro [22] proposed dYnamic and scalable tree Aware of Spatial correlaTion (YEAST) algorithm. This algorithm is almost the same as that in [21], the main difference being that the algorithm in [22] reduces the power consumption of a node by removing unnecessary broadcast alerts. The most important advantage of YEAST is the scalability of its correlation region, which it can be scaled dynamically in response to events and application constraints.

Piyare and Lee [1] have proposed a Wireless Sensor Network (WSN) connected to a cloud using the ZigBee network. They used REST-based web services to make it easier to integrate the system into any other applications for remote observation. The components of their system are: sensors integrated with a XBee ZB platform (the end device), an Ethernet router, Arduino UNO, and an Ethernet shield and the XBee ZB module. The main goal of their system is to sense the temperature of the surrounding environment and upload it to the cloud to measure power consumption. They introduced a sleep mode for the end device and measured its power consumption for two extreme cases of data packet size: 2 bytes and 101 bytes. Then, they calculated the battery lifetime for different upload periods, as shown in Figure 3-1 [1]. The results of this experiment show that Zigbee technology increased battery lifespan up to numerous years. However, the lifespan of the end device declines when the packet size expands.

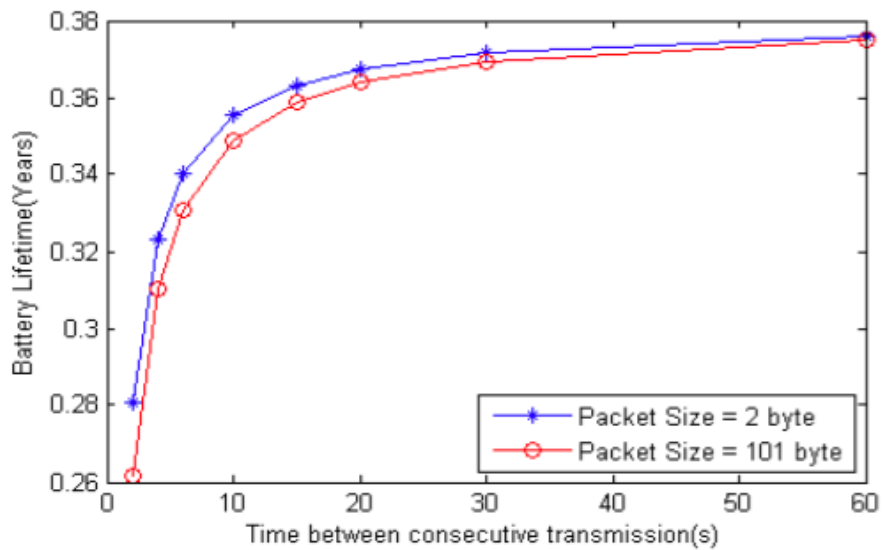


Figure 3-1: End Device Battery Lifetime for Different Packet Size [1]

In summary, we conclude that further research is needed to extend the battery life of IoT devices. Battery life has a slow growth found to be between 5% to 8% per year in the batteries with the same capacity [21]. The exciting protocols for communication between sensors and the Internet are not established to be power aware, and researchers such as [21], [22], and [23] are trying to come up with innovative ways or protocols to improve the power consumption of WSNs. Some researchers have introduced wireless communication stacks [22], while others have generated new algorithms to help IoT devices consume less power. Shabna, Jamshid, and Kumar [20] developed an STCA algorithm to eliminate unnecessary data transmission, while the authors of [22] developed YEAST to remove unneeded broadcast alerts. Both are aiming to reduce power consumption.

Although, Piyare and Lee [1] have proposed sleep mode for ZigBee network, they did not explore other technologies such as GPRS to measure the power consumption. All of the research in [1], [20], [22], and [23] is concerned about power consumption from the software side, but there is also a need for power-aware hardware.

Chapter 4

4 System Design

Our system has been designed to monitor the location, speed, and temperature of a vehicle equipped with a Tracking Unit that can connect to the Internet via the GSM/GPRS network. The Tracking Unit contains five main components: the microcontroller, GPRS/GPS model, SIM card, GPS antenna, and GSM antenna. In addition to the Tracking Unit, a cloud system is needed for storing and evaluating data. The cloud system has a web server, database, and domain name, and receives data from the Tracking Unit using the domain name and the server. Then, the web server stores the data in the database, analyzes it, and decides whether a notification should be sent, according to the thresholds set by the cloud administrator. We have also developed an Android application to receive the GPS coordinates and vehicle temperatures from the cloud, and to view notifications.

This system provides real-time tracking of location, speed, and temperature and is designed for users who want to keep track of their vehicles, such as car rental companies. For example, a rental agent will register customers in the system by adding the following customer and vehicle information: Customer Name, Phone, Email, Car Make, Car Model, Username, and Password. Then, the agent will set the location and temperature range for this particular vehicle. The customer then downloads the Android application to receive notifications. The application notifies the customer if he/she passes the allowed ranges (location or temperature). The location range is the area that the customer is allowed to drive in, such as a province. The temperature range is the maximum and minimum temperatures that are permitted inside the vehicle or its cargo hold, as set by the agent that rents this vehicle. If a customer passes from this range, an email is sent to the agency, and a notification is sent to the customer by his/her smartphone.

In general, the Tracking Unit sends the GPS coordinates and temperature to the cloud using an Arduino board and GPRS/GPS model. Then, the cloud analyzes the data and decides whether to send a notification to the smartphone. The entire system is shown schematically in Figure 4-1.

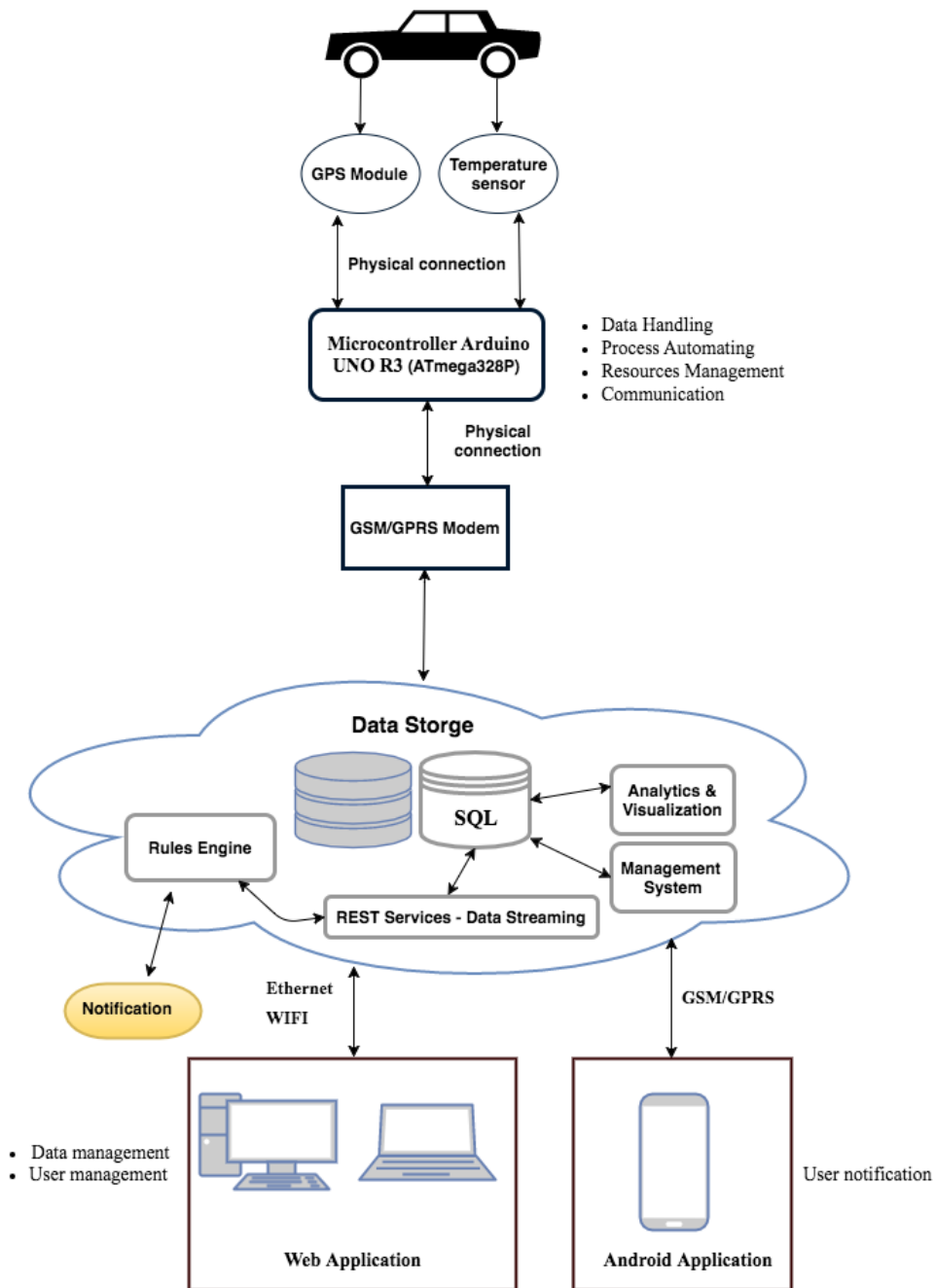


Figure 4-1: System Overview (Software Module)

For example, if the administrator has more than one vehicle with different drivers, he/she can register all vehicle and driver information on the web site, including the driver's name, phone number, and email; and the vehicle's make, model, and plate number; plus a user name and password (for the driver to use when downloading the application to a smartphone).

The administrator can set for each vehicle a location range that the driver is not allowed to pass and range of temperatures allowed inside the vehicle. The driver has to download the application to his/her smartphone in order to receive notifications. If the driver passes out of the location or temperature ranges, the administrator will be notified by email. The email will include the driver's name, date and time, speed, temperature inside the vehicle, and the warning description (has passed out of the location or temperature range). In addition, the driver will be notified of the problem details. Thus, both the administrator and the user can monitor the vehicle's use.

This system has been developed for users who own more than one vehicle, such as a car rental service or moving company. A family that has more than one vehicle also could use it. We have made the system adaptable and flexible, and it allows the user to set constraints for each vehicle, add drivers, and add vehicles. The user can also keep track of each vehicle's status by viewing the history of the vehicle on the web page.

This system uses different kinds of hardware. Some are owned by other parties such as GPS Satellites, a GSM /GPRS tower, a cloud web server and database, as shown in the Figure 4-2. The author owns the rest of hardware. The GPS satellites give precise measurements of the location of the GPS module that is connected to the microprocessor. The GSM /GPRS tower is used as a communication channel between the Tracking Unit system installed on the vehicle and the cloud web server. The cloud web server manages the data received from the Tracking Unit system. The Tracking Unit system, which contains the GPS module, temperature sensor, microcontroller, and GSM/GPRS modem, is used to sense the environment and send information to the cloud via the GSM/GPRS tower. The Android device and the personal computer are used to view, delete, and add information to the database.

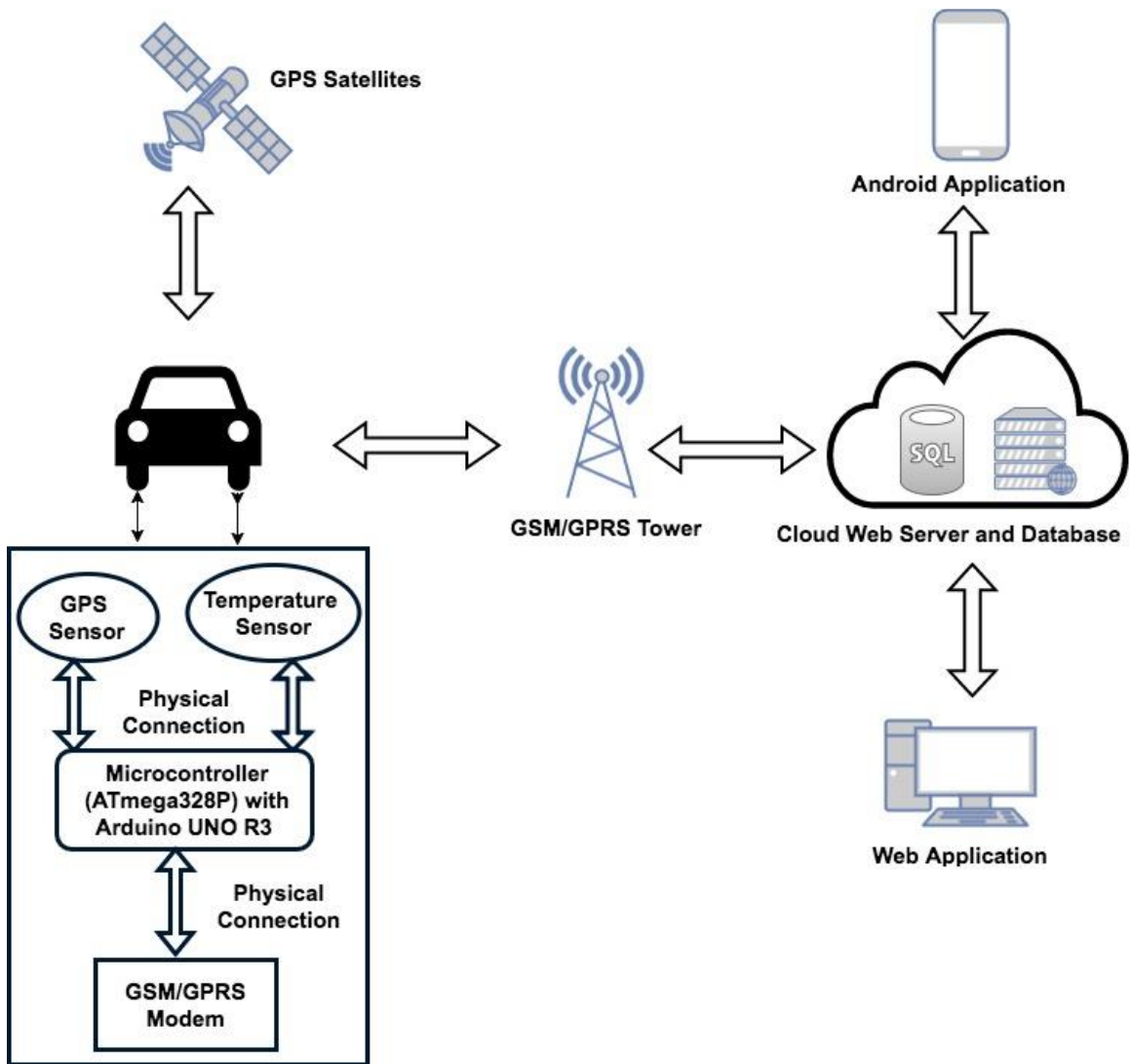


Figure 4-2: System Hardware Module

The system has three main components for obtaining the location and the temperature information: the Tracking Unit (sensors and middleware), cloud, and Android application, all explained in detail below.

4.1 Tracking Unit (Sensors and Middleware)

The Tracking Unit's three main components are the microcontroller, temperature sensor, and GPRS/GPS module (Figure 4-3).

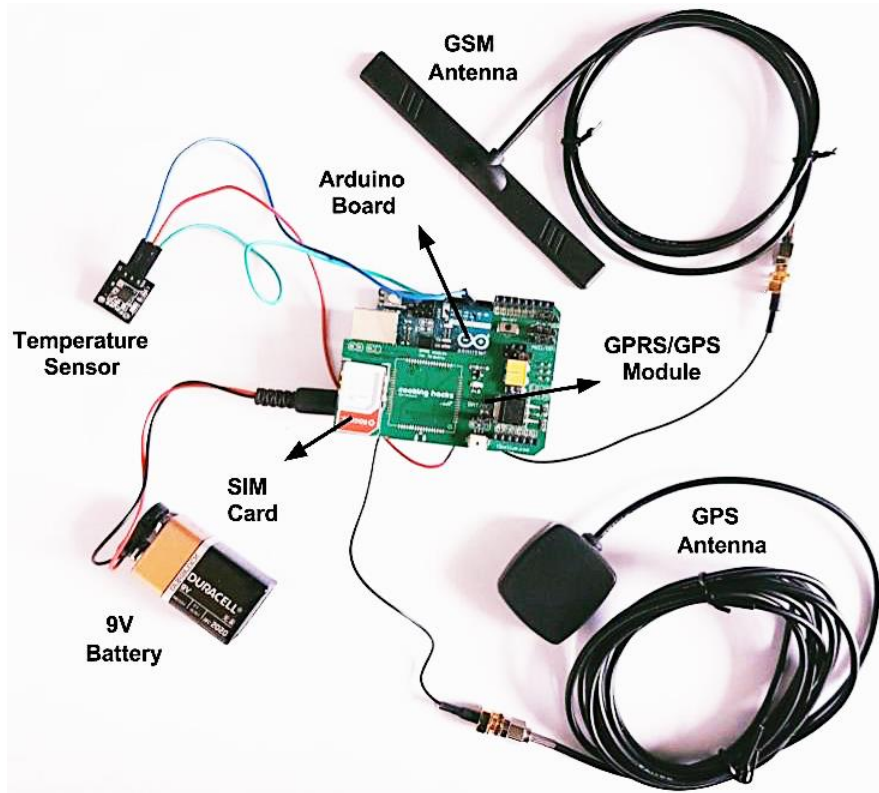


Figure 4-3: Tracking Unit Components

4.1.1 Microcontroller (Arduino UNO R3)

The Arduino UNO R3 is a microcontroller based on the ATmega328. Arduino UNO R3 is the latest version of the Arduino UNO. The Arduino board has pins for input/output, a USB Plug, a power jack (External power supply), an In-Circuit System Programming (ICSP) header, transmit LED (TX), receive LED (RX), test LED and a reset button, as shown on Figure 4-4 [24].

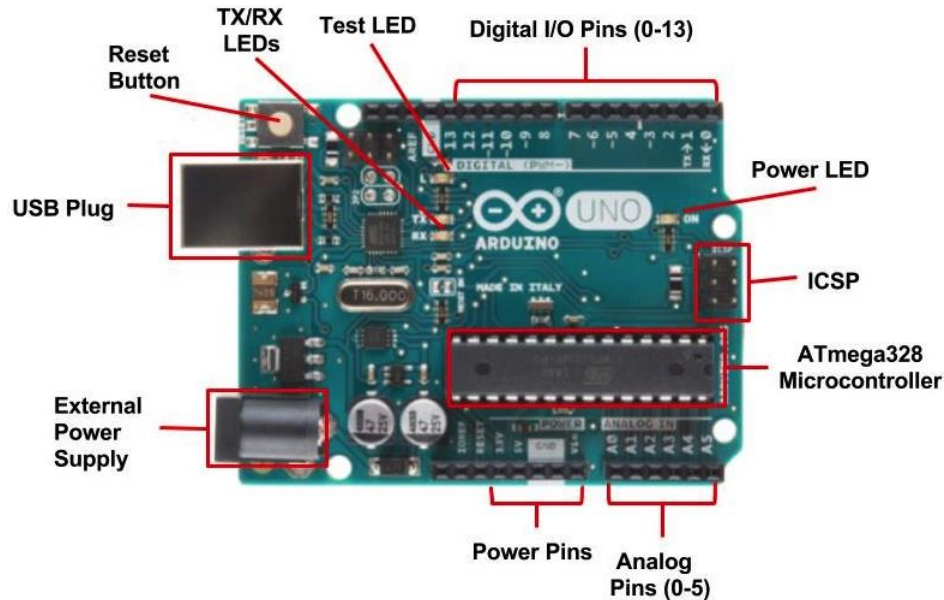


Figure 4-4: Arduino Board

The pins are divided into groups: six analog pins, 14 digital pins, and 6 power pins. The USB connection allows the Arduino board to be connected to a computer to upload codes and monitor its activities through serial communication. The Arduino Uno can be powered using the USB or AC-to-DC adapter (a 2.1mm center-positive plug) or battery, which is selected automatically by the board. Some important issues when supplying external power are as follows [24]:

- If the power supplied is less than 7 volts, the board might be unstable (i.e., it might crash or not work as it should).
- If the power supplied is more than 12 volts, the board regulator might over-heat and damage the board.
- Thus, the recommended external power supply is between 7 and 12 volts.

In this system, the power is supplied by a 9-volt battery, connected to the power jack of the Arduino board so the external power supply is between the 7 and 12 volts recommended by the manufacture.

The ATmega328 is used as the brain of the tracking system that controls the hardware used in the vehicle. An Arduino shield is installed on top of the Arduino Uno to acquire the GPS and GSM/GPRS connection. C programming language is

used as the software program to control the components of the system. The software program is compiled and saved in the microcontroller's flash memory.

Specifications:

- Microcontroller ATmega328
- Operating Voltage 5V
- Input Voltage (limits) 6-20V
- Digital I/O Pins 14 (of which 6 provide PWM output)
- Analog Input Pins 6
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32 KB of which 0.5 KB used by bootloader
- SRAM 2 KB
- EEPROM 1 KB
- Clock Speed 16 MHz

4.1.2 Temperature Sensor (DS18B20)

The temperature sensor (DS18B20) is used on a board called the Keyes board as shown in Figure 4-5, which has been designed to work on many microcontrollers such as the Arduino and Raspberry. For this system, the sensor is used with an Arduino board.

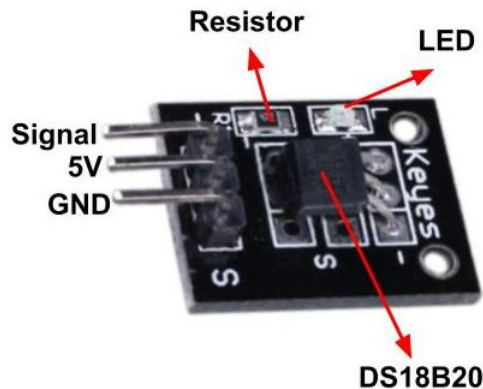


Figure 4-5: Temperature Sensor

The temperature sensor board has three components: the DS18B20, flashing LED, and resistor. The DS18B20 is a digital thermometer. The flashing LED is used to indicate whether the temperature board is powered correctly or not. If there is a problem with the power, then the light will dim. To control or limit the flow of electric current from the

Arduino board, a resistor has been used, with a library called OneWire because this sensor uses a one-wire bus. To connect the sensor, three pins of the Arduino board are used: DS(10) (to transmit the data), GND (to connect to ground), and V5 (to get 5-volt power). We have used three wires to connect the temperature board to the Arduino board (Figure 4-6). The orange wire is used to get the power from the V5 pin on the Arduino board. The green wire is used as a ground and connected to the GND pin in the Arduino board. The blue wire is used to transmit data from the temperature sensor to the Arduino board using digital pin number 10. On the software side, we used DS(10) to create an object of class OneWire (of the library OneWire.h). The number 10 represents the digital pin on which the temperature sensor data is sent to the Arduino board.

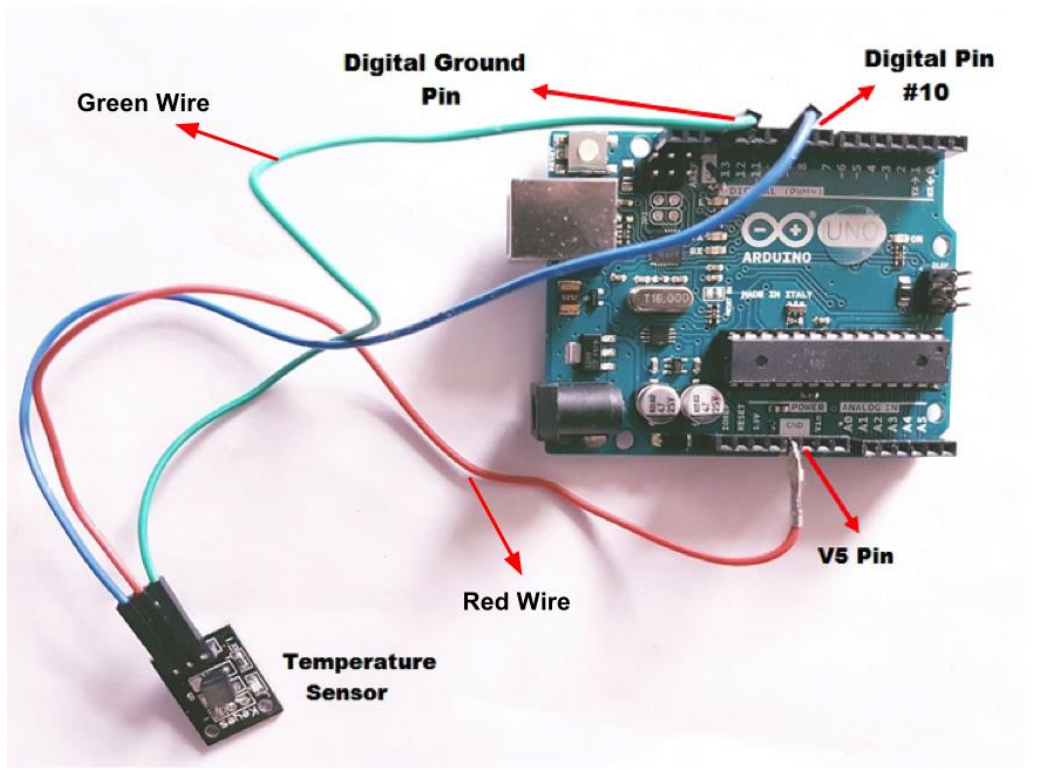


Figure 4-6: Arduino and Temperature Sensor Physical Connection

Specifications [25]:

- The sensor can measure temperatures from -55 to +125 Celsius (-67 to +257 Fahrenheit).
- Accuracy of the temperature sensor is ± 0.5 Celsius.
- Uses from 9 bits to 12 bits.

4.1.3 GPRS/GPS Quadband Module (SIM908)

This module or shield is designed to work on many microcontrollers, such as the Arduino and Raspberry Pi. For this system, the shield is used with an Arduino Uno. The shield includes the SIM908 module to connect to the network via GPRS/GSM and to get the GPS coordinates (longitude and latitude). It is deployed as a communication medium between the Arduino and the web server due to its wide coverage and low cost [26]. The GPRS/GPS Quadband Module communicates with the Arduino board via the UART port of the Arduino on pins D 0 and D 1, as shown in Figure 4-7. Pin D 2 is used as a reset switch pin to power the shield. The In-Circuit Serial Programming (ICSP) Pins are used to program the ATmega328 microcontroller and the GPS/GPRS module after its installation, rather than programming it prior to installation on the Arduino board [25] [33]. The rest of the pins (D 3-7 and A 0-5) are connected to the Arduino board as an extension for more convenience if more connections are needed in the future.

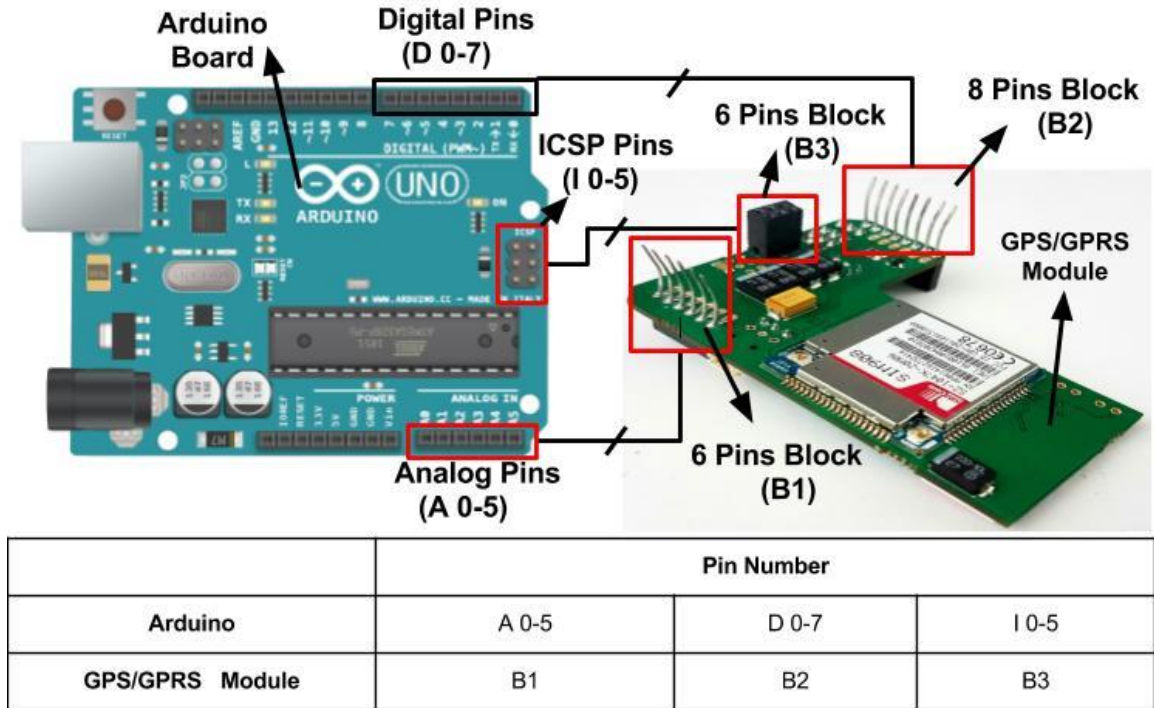


Figure 4-7: Physical Connection Between Arduino and GPS/GPRS Module

The GPS/GPRS shield is used for real-time tracking systems that obtain GPS coordinates and send them to a web server using Hyper Text Transfer Protocol (HTTP) requests. The SIM908 is connected to a GPS antenna and GSM antenna, as explained in detail in Figure 4-11. There are three ways to power this board: external power, the Arduino, or a battery [27]. If one changes the power source, the jumpers on the board must also be changed, as explained in Table 1 and shown in Figure 4-8 and Figure 4-9. In this project, the power comes from the Arduino board.

Table 1: Jumpers Arrangement for GPRS/GPS to be Power from Arduino Board

Power Source	Charger Jumper	Vin Jumper	REG/BAT Jumper
External Power	Remove Jumper	Move jumper to Vext position	Move jumper to REG position
From Arduino	Remove Jumper	Move jumper to Ard position	Move jumper to REG position
From Battery	Put Charger Jumper	Remove Jumper	Move jumper to BAT position

Specifications [28]:

- Quad Band 850 / 900 / 1800 / 1900 MHz
- GPRS Multi-slot class 10
- GPRS Mobile Station class B
- Compliant to GSM phase 2 / 2+ - Class 4 (2W @ 850 / 900 MHz) - Class 1 (1W @ 1800 / 1900 MHz)
- Dimensions: 30 x 30 x 3.2mm
- Weight: 5.2g -SIM908:11.1g
- Control via AT Commands
- SIM Application Toolkit
- Supply Voltage Range -GPRS: 3.2V ~ 4.8V -GPS: 3.0V ~ 4.5V
- Low Power Consumption
- Normal Operating temperature: - 40°C to +85°C
- GPRS class 8/10: max. 85.6 kbps (downlink)
- Coding schemes CS 1, 2, 3, 4
- Integrated TCP/IP stack

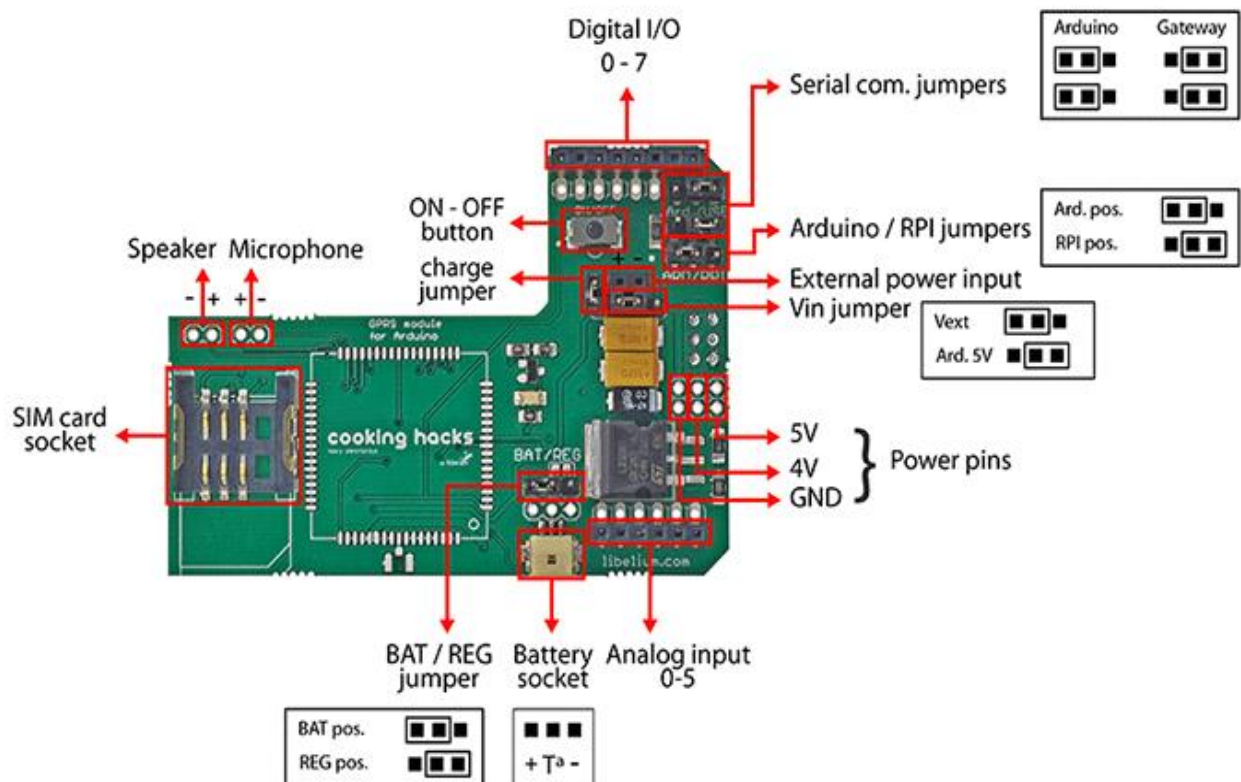


Figure 4-8: GPRS/GPS Shield Diagram (Face Up):

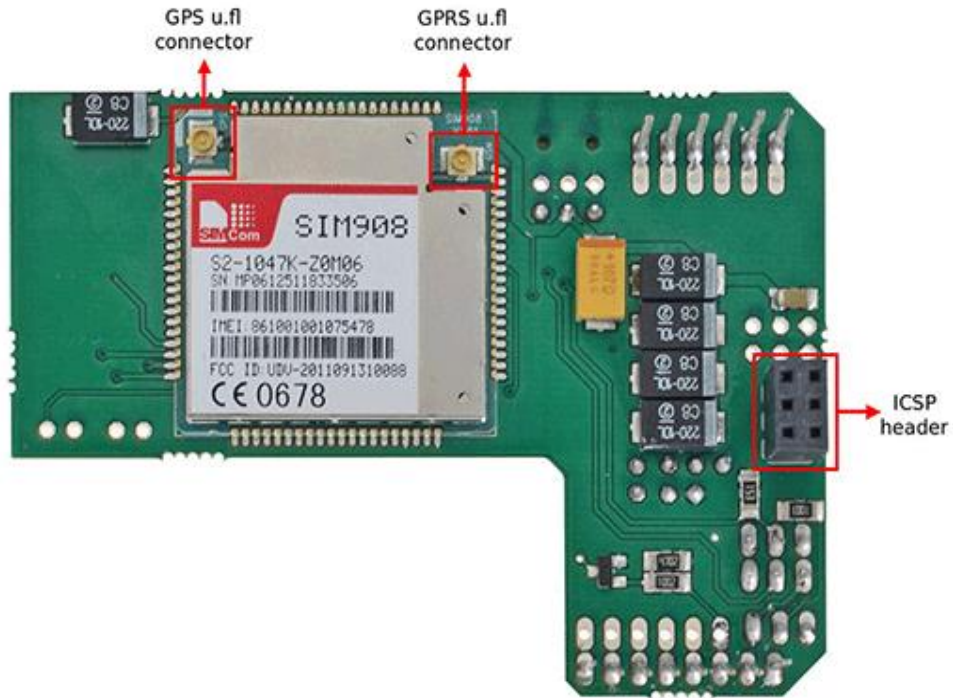


Figure 4-9: GPRS/GPS Shield Diagram (Face Down)

4.1.4 GPS Antenna

This system uses an external GPS antenna to connect to satellites, as shown in Figure 4-10. This antenna is connected to the SIM908 to give the exact position of the antenna, speed of the vehicle, and how many satellites are connected to it. A magnet is used to make putting the antenna on top of a vehicle easier. To connect to a satellite, the GPS antenna has to have a clear view of the sky because of the weak capability of the antenna. A connector from the SMA-Female to the UFL pigtail connects the GPS antenna to the SIM908, as shown in Figure 4-11 below. This GPS receiver receives location information from the satellite, and then the software decodes it using the National Marine Electronics Association (NMEA) protocol.



Figure 4-10: GPS Antenna

Specifications [29]:

- Frequency: GPS 1575.42 MHz
- Impedance: 50 Ohms
- Supply Voltage 2.7V - 5.5V
- Current: 15mA - 25mA
- Power (max.): 125mW
- Connector: SMA Male
- Operating temperature -40°C to +85°C

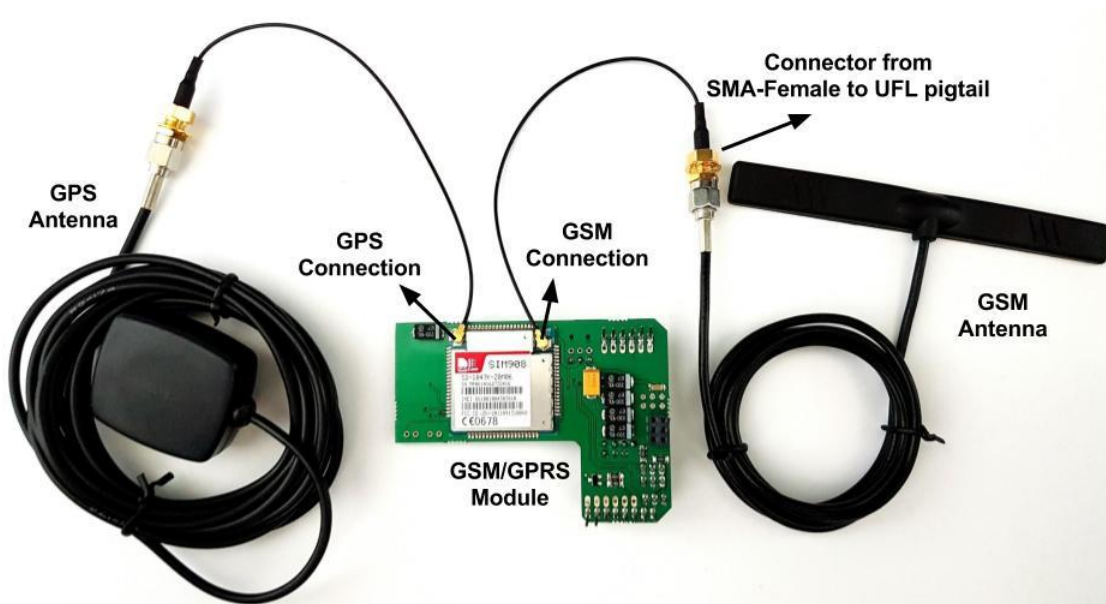


Figure 4-11: Connection of GPS and GSM/GPRS Antennas with GPS/GPRS Module

4.1.5 GSM/GPRS Antenna

An external GSM/GPRS antenna is connected to the SIM908 to provide Internet connection through the GSM/GPRS network, as shown in Figure 4-12. A connector from the RPSMA -Female to UFL pigtail is used to connect the GSM/GPRS antenna to the SIM908, as shown above in Figure 4-11. This antenna works on LTE, 4G, 3G, and GSM frequencies.



Figure 4-12: GSM Antenna

Specifications [30]:

- Frequency: 4G/LTE (791-862/1710-2690 MHz), 3G (UMTS 2.1 GHz), GSM Quadband (850-900-1800-1900 MHz)
- Impedance: 50 Ohms
- Polarization: horizontal
- Gain: 2.14 dBi
- VSWR: < 2:1
- Power handling: 25 W
- Connector: RPSMA Male
- Operating temperature: -40°C to +85°C

4.1.6 9V Battery and Barrel Jack Adapter

The battery used for this system is a Duracell alkaline 9V, which has a capacity of 550 mAh, IEC: 6LR61, ANSI: 1604A, and nominal 9 voltage. This battery makes the Tracking Unit system portable and able to be positioned in any vehicle without installation. To connect the battery to the microcontroller (Arduino), we used a 9V

battery connector as shown in Figure 4-13. This battery powers the Arduino and the GSM/GPRS module at the same time because we have changed the setting of the GSM/GPRS module to be powered from the Arduino.



Figure 4-13: 9V Battery and its connector

4.1.7 Tracking Unit Communication

C language is used in the Integrated Development Environment (IDE), and the IDE is open source and can be downloaded from the Arduino website. It gives the user the ability to create and edit sketches that can then be uploaded to the Arduino board using a USB cable. To develop a code in the IDE, two main function are needed: `setup()` and `loop()`. These two functions are defined as void, so there is no return value. The setup function is used to initialize the setting and is called once in the start of the application. The loop function is constantly called during the execution of the application [30].

To set up the communication, a USB connection between the Arduino and a Personal Computer (PC) is needed. This connection gives the ability to upload code to Arduino and monitor its activities though serial communication. The communication between the PC and the Arduino occurs through AT commands using a baud rate of 115200.

The Tracking Unit goes through three different phases. The first starts when the serial communication is open. The code checks if the GPRS/GPS

module is connected or not. Then, the Subscriber Identity Module (SIM) card's Personal Identification Number (PIN) is sent and starts powering and acquiring GPS signals until it is connected to the satellites, as shown in Figure 4-14.

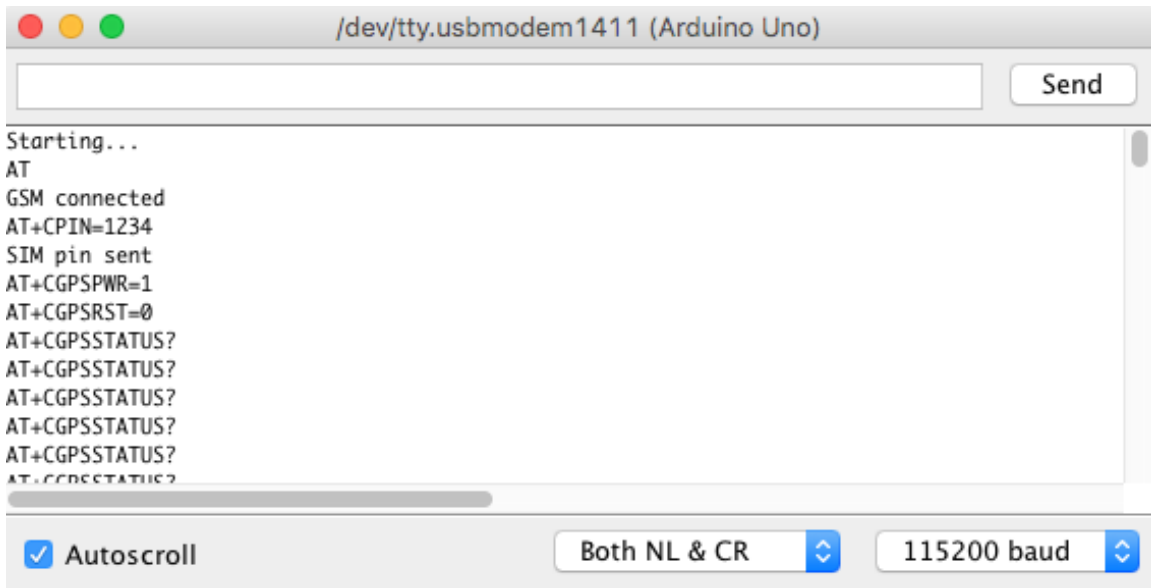
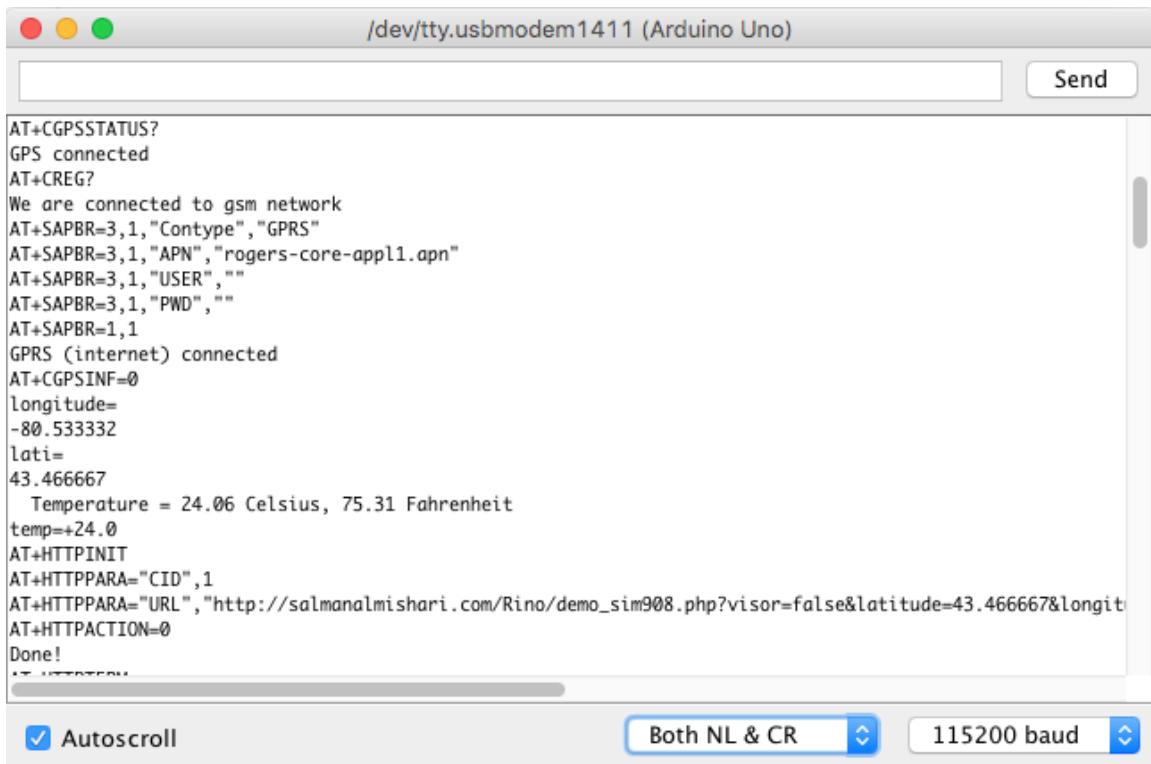


Figure 4-14: Tracking Unit System Code in Phase 1

The second phase deals with network registration, GPS information, and temperature reading. First, the code makes sure the SIM908 is registered to a network by using the AT+CREG command. Then, the information for connecting to the GPRS is sent, including the Access Point Name (APN), username, and password. After that, an AT+CGPSINF command is used to obtain the latitude, longitude, altitude, UTC time, speed, course, and number of connected satellites. After getting the GPS information, the temperature is acquired from the temperature sensor using the one-wire library. The temperature sensor is connected to the Arduino board using three pins: D10, GND, and 5V. D10 is a digital pin, and is used to transfer data from the sensor to the Arduino. The GND and the 5V pins are used to power the temperature sensor. At this point, all the information needed is stored in variables in the Arduino memory in case of network disconnection. Then, an HTTP connection is initialized, and all information is sent using GET method, which sends all the information needed to the web

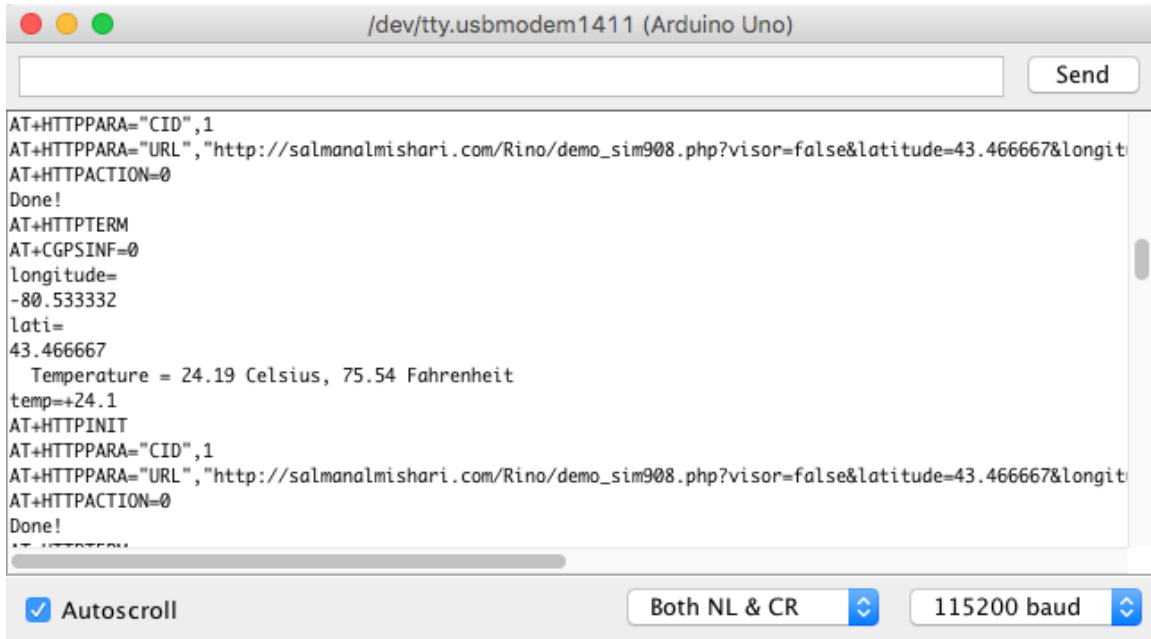
server to be stored and analyzed, as shown in Figure 4-15. Phase 1 and phase 2 require up to one minute to warm up and create all the necessary connections.



```
AT+CGPSSTATUS?  
GPS connected  
AT+CREG?  
We are connected to gsm network  
AT+SAPBR=3,1,"Contype","GPRS"  
AT+SAPBR=3,1,"APN","rogers-core-appl1.apn"  
AT+SAPBR=3,1,"USER",""  
AT+SAPBR=3,1,"PWD",""  
AT+SAPBR=1,1  
GPRS (internet) connected  
AT+CGPSINF=0  
Longitude=  
-80.533332  
Latitude=  
43.466667  
Temperature = 24.06 Celsius, 75.31 Fahrenheit  
temp=+24.0  
AT+HTTPINIT  
AT+HTTPPARA="CID",1  
AT+HTTPPARA="URL","http://salmanalmishari.com/Rino/demo_sim908.php?visor=false&latitude=43.466667&longit  
AT+HTTPACTION=0  
Done!  
AT+HTTPDATA
```

Figure 4-15: Tracking Unit System Code in Phase 2

The third phase is continually repeated to acquire GPS and temperature information, and then create output messages in JavaScript Object Notation (JSON) format and send them through the HTTP protocol to the web server. This phase can be repeated every second so that real-time tracking is achieved, as shown in Figure 4-16.



```
/dev/tty.usbmodem1411 (Arduino Uno)
Send
AT+HTTPPARA="CID",1
AT+HTTPPARA="URL","http://salmanalmishari.com/Rino/demo_sim908.php?visor=false&latitude=43.466667&longit
AT+HTTPACTION=0
Done!
AT+HTTPTERM
AT+CGPSINF=0
longitude=
-80.533332
lati=
43.466667
Temperature = 24.19 Celsius, 75.54 Fahrenheit
temp=+24.1
AT+HTTPINIT
AT+HTTPPARA="CID",1
AT+HTTPPARA="URL","http://salmanalmishari.com/Rino/demo_sim908.php?visor=false&latitude=43.466667&longit
AT+HTTPACTION=0
Done!
AT+HTTPTERM
```

Autoscroll Both NL & CR 115200 baud

Figure 4-16: Tracking Unit System Code in Phase 3

Figure 4-17 is a flow chart showing the Tracking Unit’s communication and processing steps in their entirety. First, step-by-step clarification of the code process is provided below.

Step (1): When the Arduino board starts up, the code sets the UART0 at a baud rate of 115200 for serial communication between the PC and the Arduino board.

Step (2)(3): The code checks whether the serial port is available or not. If available, it goes to Step (4). If not, it checks again until the port is available. This check is used to make sure that the serial communication between the PC and the Arduino board is possible.

Step (4): the microcontroller sends an AT command to the GPRS/GPS module to make sure it is connected to the Arduino board.

Step (5)(6)(7): If the GPRS/GPS module is connected, the response is “OK”. If not, the Arduino’s processor waits for a specific period and tries again.

Step (8): After checking that the GPRS/GPS module is connected, the SIM card PIN is sent so that the SIM card can be unlocked.

Step (9): The code then turns on the GPS receiver on the GPRS/GPS module and acquires a signal from the satellites.

Step (10) (11): The microcontroller checks whether the GPRS/GPS module is connected to the satellites. If not, it waits for two seconds and then sends an AT command to try to connect again.

Step (12): After the GPS is connected, the code registers the GPRS/GPS module to the GSM/GPRS network, which in this case is Rogers as a network provider.

Step (13) (14): The microcontroller checks that the GPRS/GPS module has been registered and connected. After connecting the SIM card, it sends APN info and checks whether an Internet connection is established. If not, it waits for two seconds and then tries to connect again.

Step (15): Requests the temperature and the location values from the sensors.

Step (16) (17): Extracts the information needed from the package that has been sent from the satellites, such as location (longitude, latitude), speed, date and time, number of satellites connected. Then, it stores them as values on the Arduino board. After that, all the data stored in the Arduino board is uploaded to the cloud and stored in the database.

Step (18): The microcontroller waits for specific time set by the administrator and then repeats steps 15, 16, and 18 every specified time so the location and the temperature values are updated and uploaded to the database on the cloud continuously.

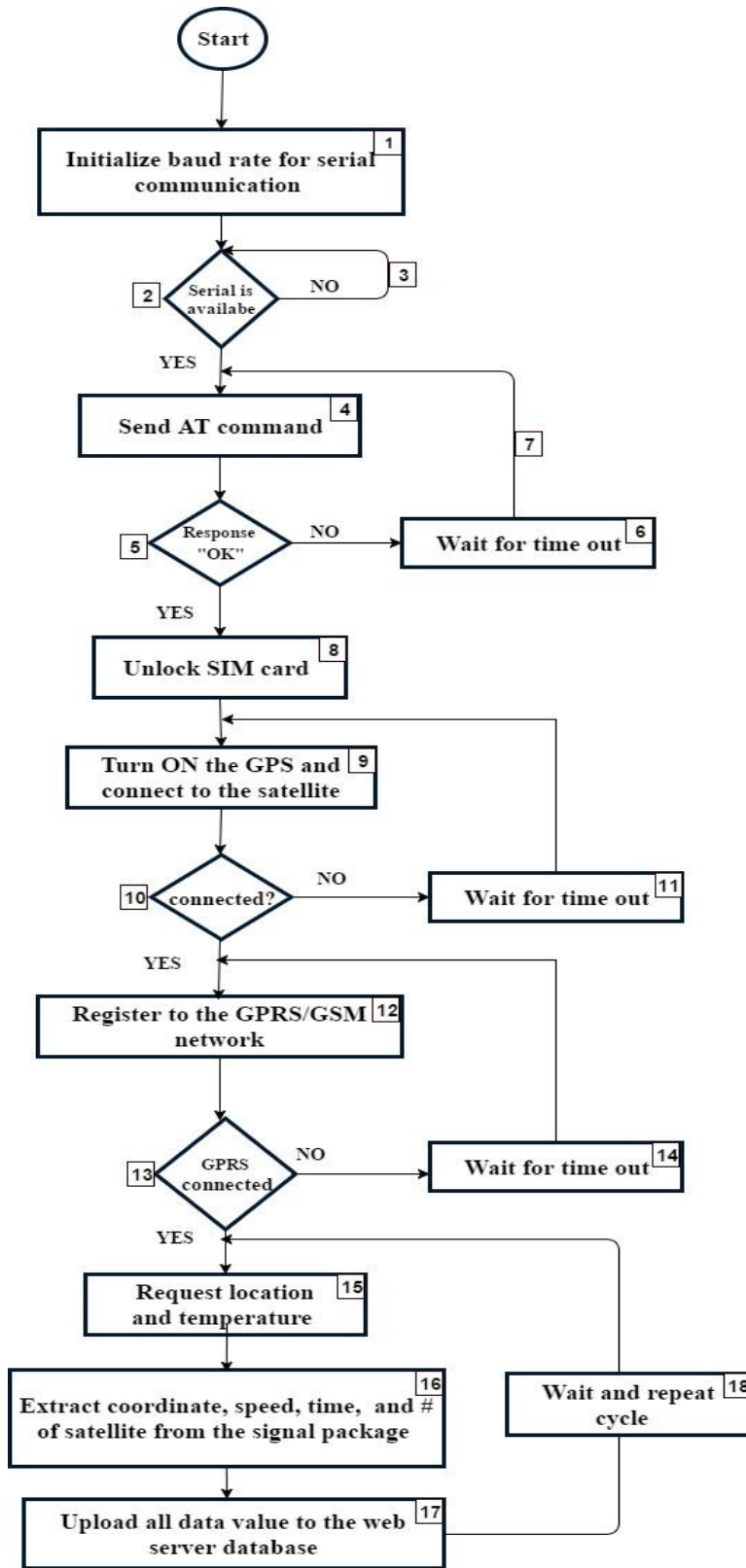


Figure 4-17: Tracking Unit Communication Flow Chart

4.2 Cloud

The cloud uses a website that will run, store, process, analyze, and manage the data. This means that the sever and database are in the cloud and an Internet connection is needed. To build a cloud, a web host and a domain name are required so the cloud can be accessed from anywhere.

4.2.1 Web Host and Domain Name

The HostGator web-hosting service has been used to develop a cloud base for the system, and includes the web panel, web server, and database. The subscription package is called Baby Croc. The server uses the Linux Cent Operating System and SQL database.

We use Domain Name System (DNS) to reach our developed cloud, and it works as an identification for the Internet Protocol address (IP address) of the location of the server on the World Wide Web (WWW). The domain name of this system is SalmanAlmishari.com, which has been purchased from GoDaddy, an Internet-domain registrar and web-hosting company.

4.2.2 Website Design

The website has been designed and developed using PHP, HTML, CSS, Ajax, XML, MySQL and JavaScript language. The web design includes a flexible architecture for integration of IoT systems such as SVS to the cloud for data gathering and sharing using RESTful web services. These services act as a compatible application for client/server architecture, which are considered to use stateless communication protocols. HTTP was used in our system for the purpose of stateless communication. The website architecture contains six main pages: Login, Home, Details, Edit, Event, Temperature and Temperature Range page. Each page is explained in more detail next.

4.2.2.1 Login Page

The Login page is used to get the credential of an administrator user wanting to access the website, as shown in Figure 4-18. In addition, it has the ability to reset passwords if forgotten.

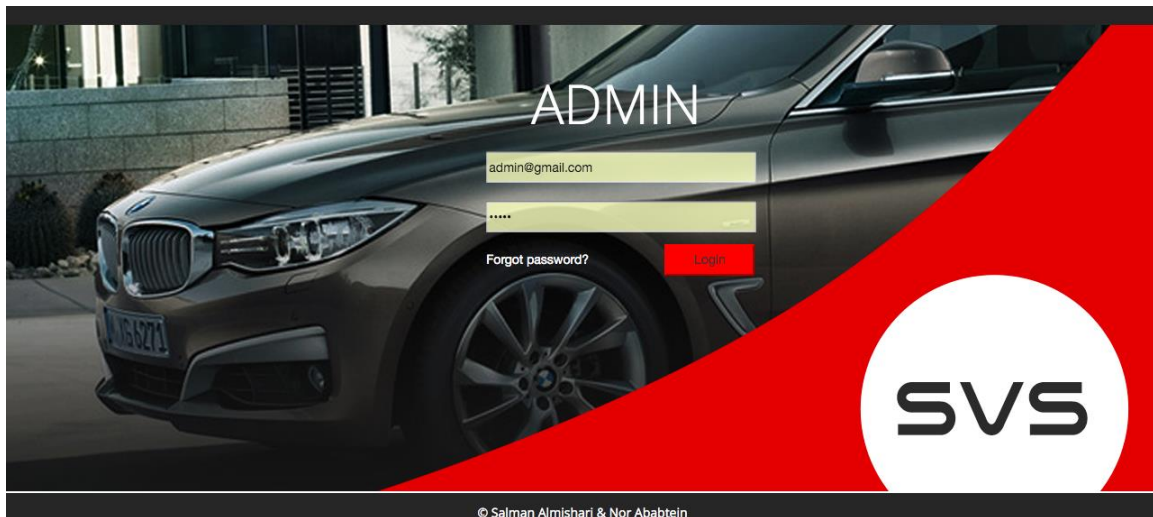


Figure 4-18: Login Page

4.2.2.2 Home Page

The home page contains a list of the entire vehicle's registered drivers, plus the make, model, and temperature inside the vehicle. Furthermore, it gives the user the ability to access the user and vehicle details, edit the vehicle's range, and view the vehicle's history. In addition, the home page has a menu bar that offers users an opening to logout, register new users, and set alerts, as shown in Figure 4-19.

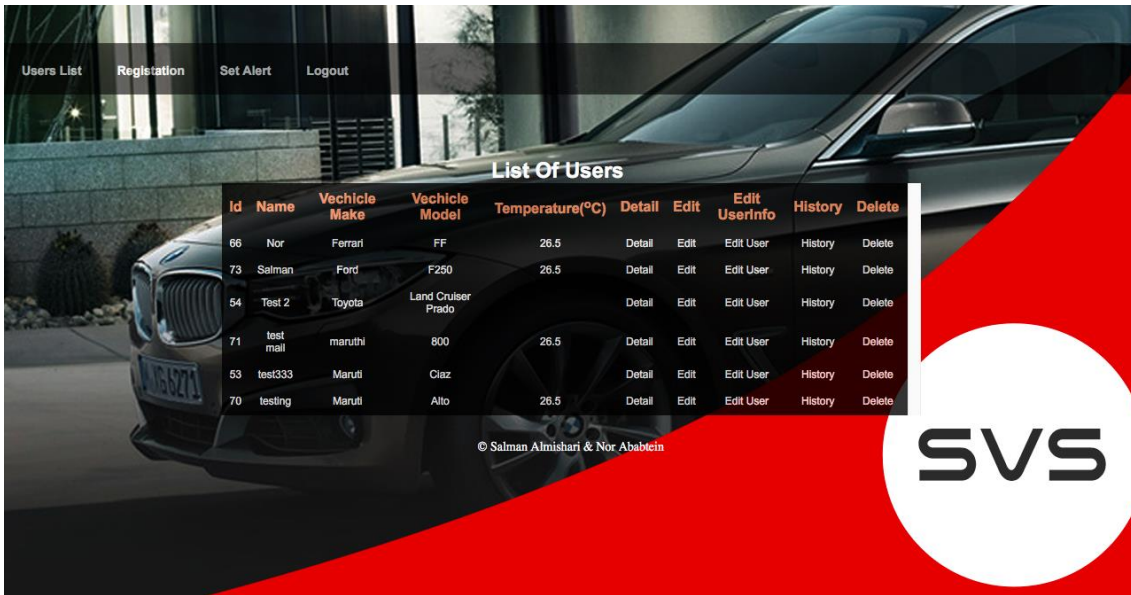


Figure 4-19: Home Page

4.2.2.3 Details Page

As shown in Figure 4-20, the detail page has all the driver information such as the name, email, phone number, plus the plate number of the vehicle used, and its current temperature, location, and which tracking device has been used. It also includes an embedded Google map that shows the vehicle location. In this page, the administrator can set the location of the vehicle for testing, or it could get the location from a GPS that has been installed in the vehicle.

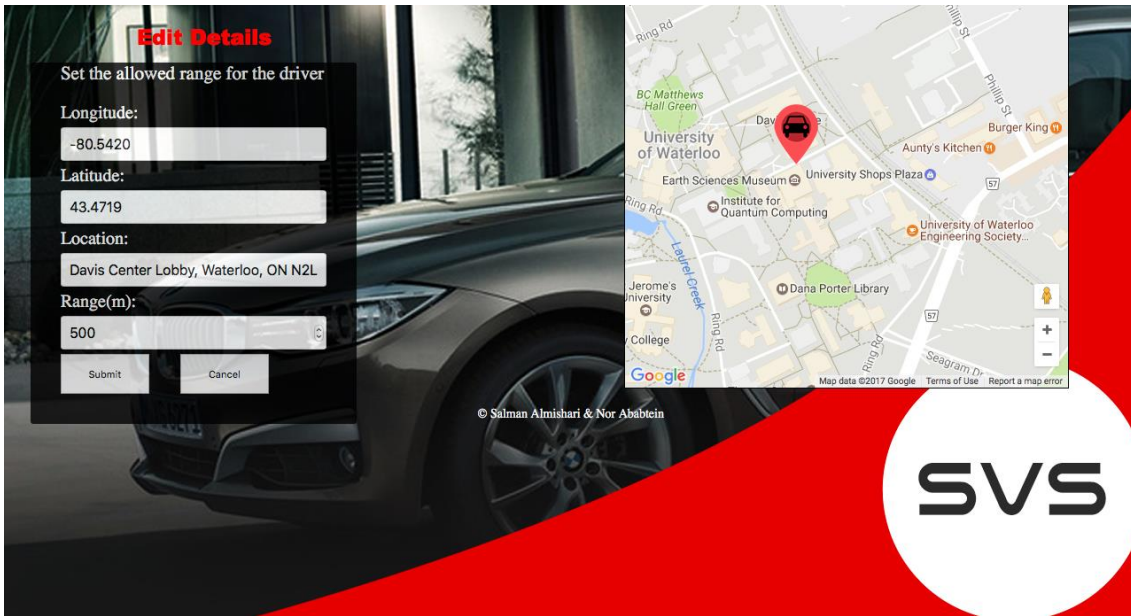


Figure 4-20: Details Page

4.2.2.4 Edit Details Page

The edit detail page (Figure 4-21) gives the administrator the ability to set the radius range from a specific location. The Google map shows the vehicle's exact location.

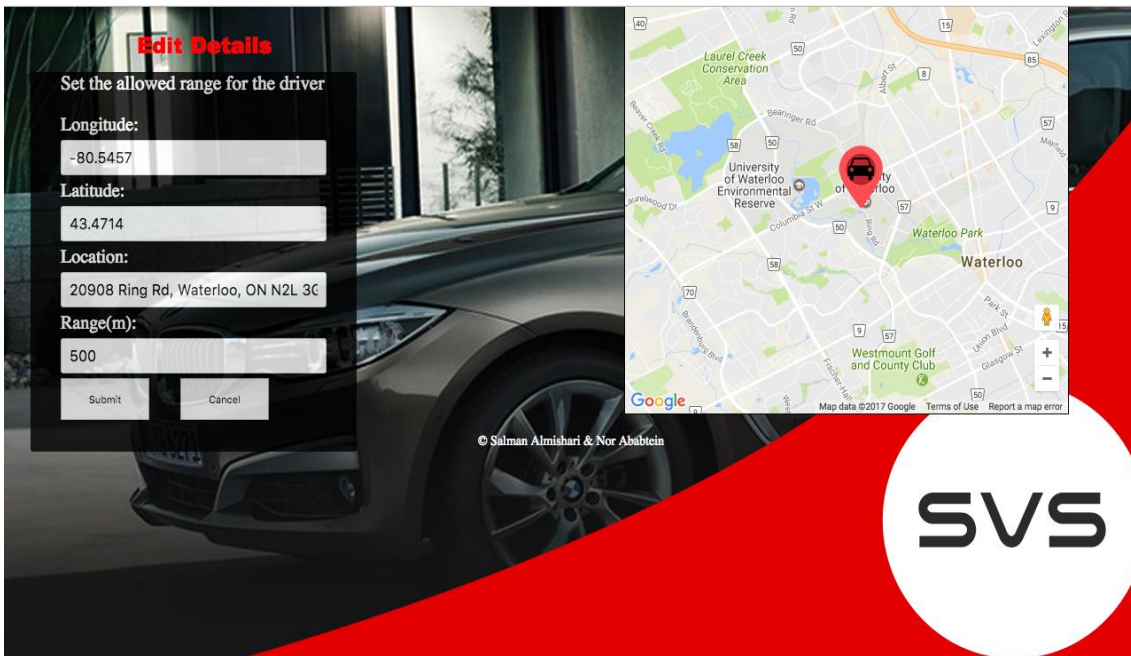


Figure 4-21: Edit Details Page

4.2.2.5 Edit User Information Page

This page (Figure 4-22) gives the administrator the ability to edit the user's information: the name, email, username, password, phone number, car make and model, the id of the device used inside the vehicle, and the vehicle's plate number.

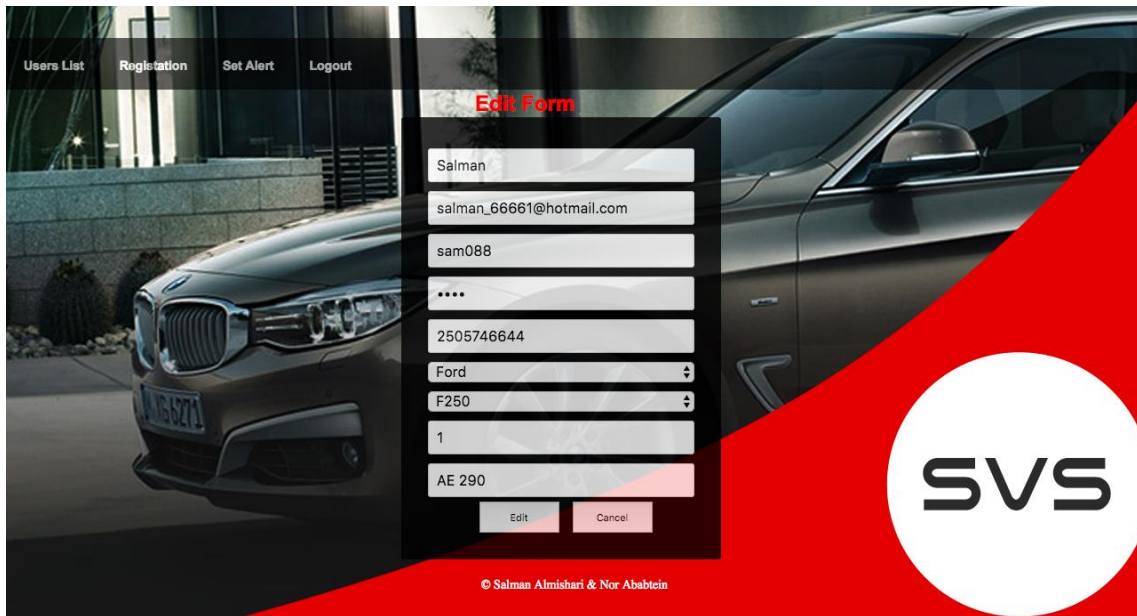


Figure 4-22: Edit User Information Page

4.2.2.6 History Page

This page (Figure 4-23) gives the past and current location, date and time, temperature, and speed of a vehicle and so provides the administrator with a record of the vehicle. For example, we have registered a vehicle and a driver, and then installed a Tracking Unit and driven the vehicle through Waterloo and Kitchener, Ontario. The Tracking Unit sends longitude and latitude of the vehicle, and then we used an embedded code to convert them to a location name. The location and speed of our route, plus the temperature inside the vehicle, was stored in the database, and this page (Figure 4-22) shows the accurate information arranged by the date and time received.

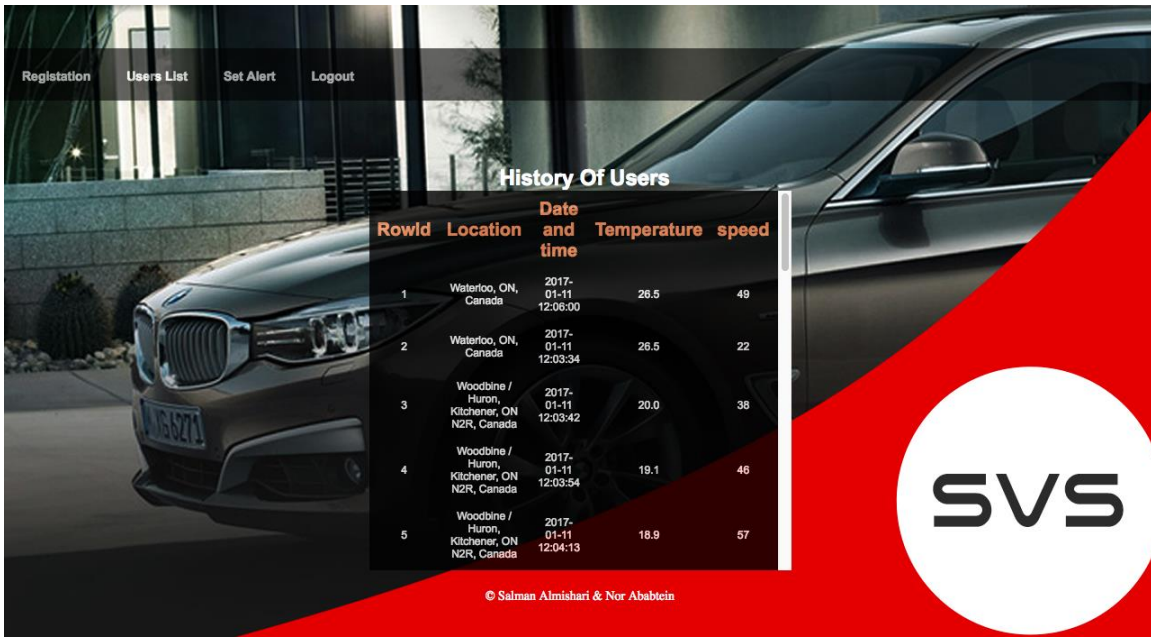


Figure 4-23: History Page

4.2.2.7 Temperature Page

The Temperature page lists all the vehicles, with their ID, make, model, and highest and lowest permissible temperatures. Therefore, each vehicle will have a range of temperatures that can be set by the administrator. If the temperature passes out of this range, then a notification trigger will be issued, as shown in Figure 4-24.

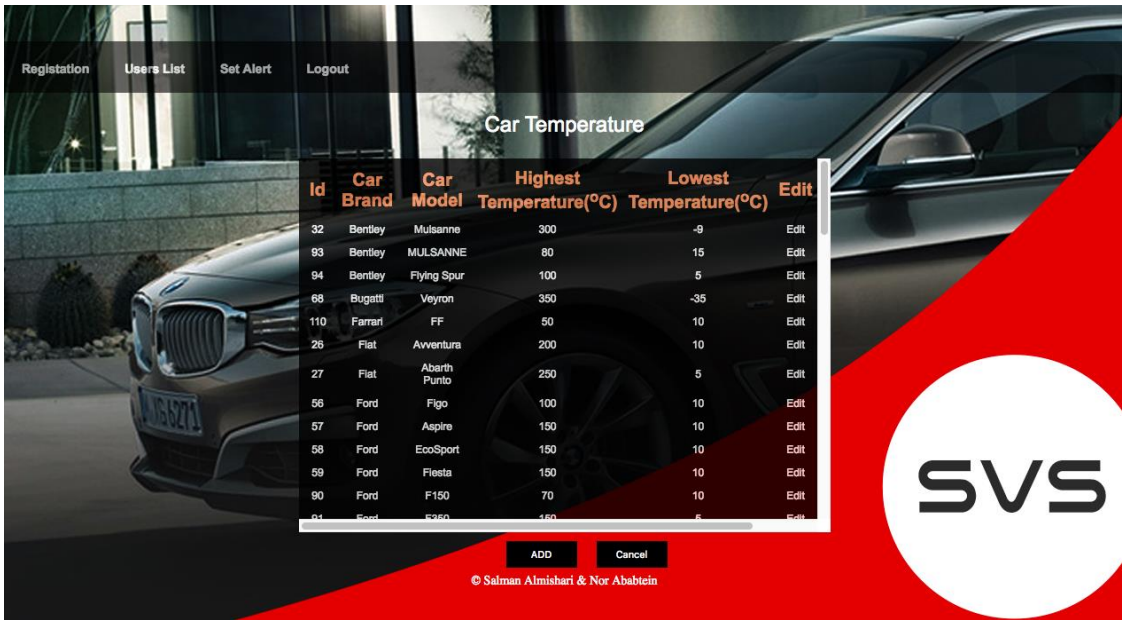


Figure 4-24: Temperature Page

4.2.2.8 Edit Temperature Range Page

This page (Figure 4-25) gives the administrator the ability to set the highest and the lowest permissible temperatures for the vehicle, so when the system senses that a reading is above or below range, a notification will be sent to the smartphone followed by email to the administrator. The administrator has to choose the vehicle's make and model, and then he/she can set the highest, lowest, or both temperatures.

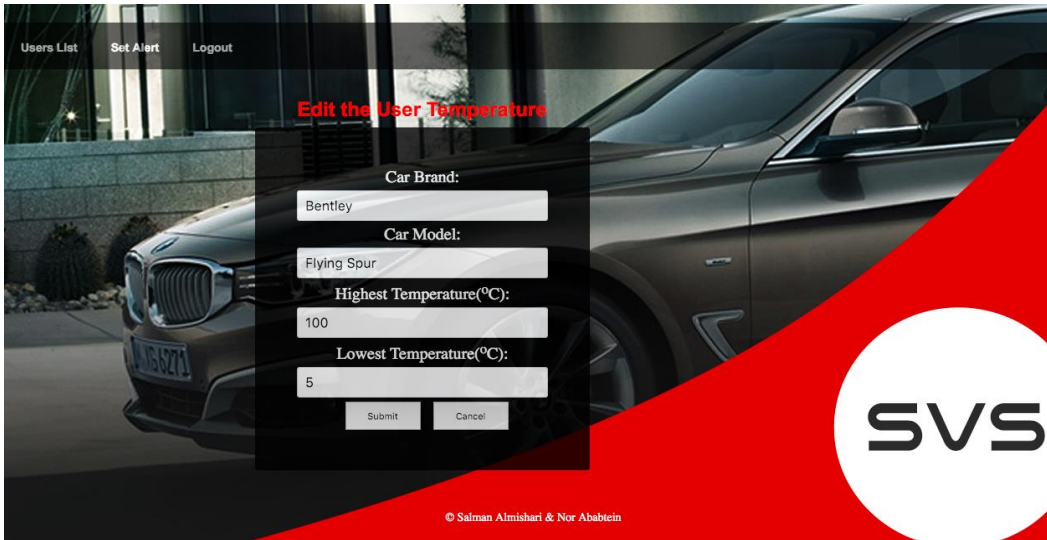


Figure 4-25: Edit Temperature Range Page

4.2.2.9 Event Page

The Event page gives the administrator the ability to broadcast to all the drivers. This feature has been added for events like gas discounts or new policies that are imposed by the company. As shown in Figure 4-26, the administrator inserts the event name and description then submits it. Then, all of this information is sent to all the drivers registered on the system.

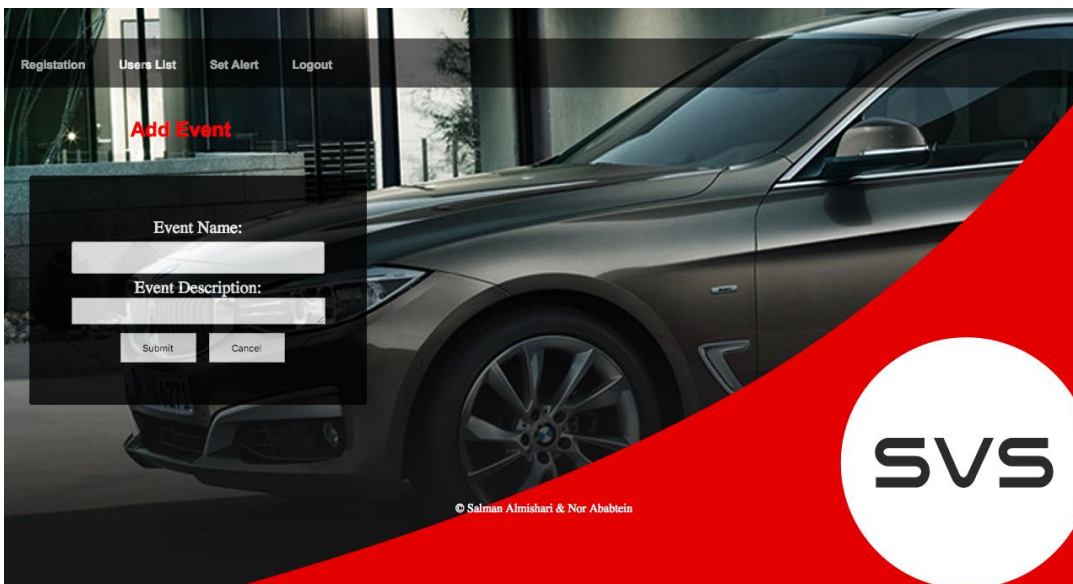


Figure 4-26: Event Page

4.3 Android Application

The Android application has been developed using Java language and the Eclipse development environment. The Android application has four main pages: the login, home, profile, and about us page.

4.3.1 Login Page

The Login page for the Android application is used to check the credential of a driver wanting to access the application, and for fetching his/her vehicle's information. This access requires a username and password that are provided by the administrator. When the administrator inserts the vehicle and driver information, he/she is asked to create a username and password for the driver, and then the driver can update or change the password after he/she has accessed the application. The username and password information are stored on the cloud, and a credential check is preformed when the information is submitted, as shown in Figure 4-27.

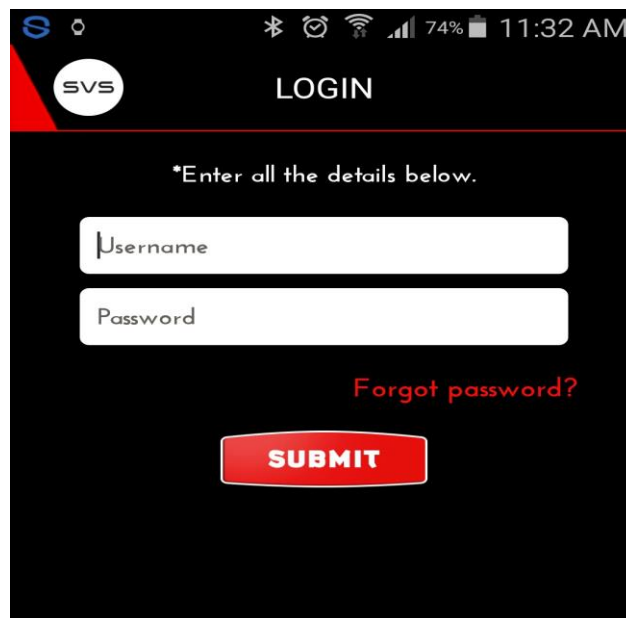


Figure 4-27: Android Application Login Page

4.3.2 Home Page

The home page contains the most important information. Every time this page is accessed, the current vehicle's data is presented, as shown in Figure 4-28. This data is fetched from the Cloud, which contains the most recent records uploaded from the Tracking Unit. The presented data is listed below:

- The current temperature of the vehicle.
- The temperature range that was set by the administrator.
- The current latitude and longitude of the location of the car.
- The current city, province, and country of the car.
- An embedded Google map to show the exact location.

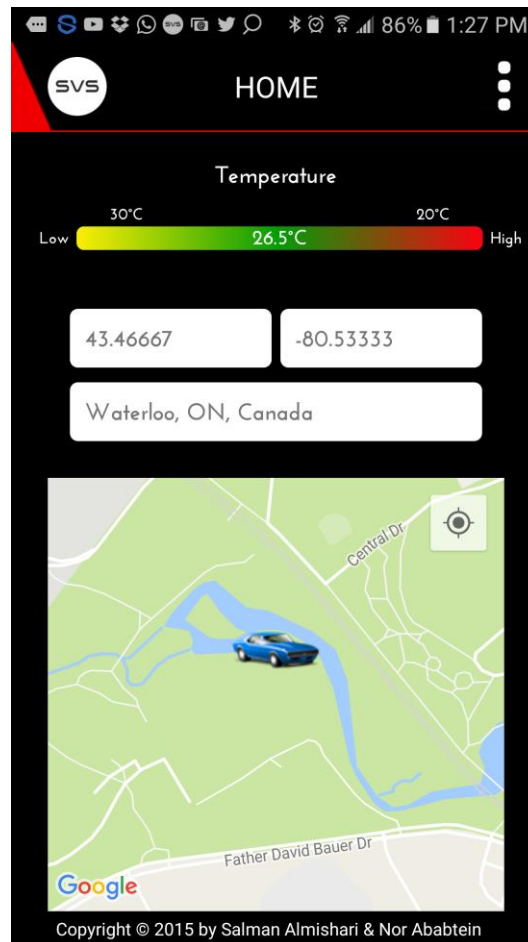


Figure 4-28: Android Application Home Page

4.3.3 Profile Page

The profile page contains the driver's info, vehicle's make and model, and the username. In addition, the user can update his/her info or change the access password to the Android application, which is directly linked to the database on the Cloud. Figure 4-29 provides an example of this page, using the following data:

- 1- The name of the driver, which is Salman.
- 2- The driver's email: salman_66661@hotmail.com.
- 3- The driver's phone number: 2505746644
- 4- The make and model of the car that is used by this particular driver: Ford, F250
- 5- The username that has been given to the driver.
- 6- The new password if the driver wants to change the current password.

UPDATE PROFILE

Update details below.

Salman

salman_66661@hotmail.com

2505746644

Ford

F250

sam088

Password

Confirm Password

SUBMIT

Figure 4-29: Android Application Profile Page

4.3.4 About Page

The “About” us page contains a description of the application, the development version, and the authors of the developed application. This page gives drivers some information about the application and the main functionality that has been implemented, as shown in Figure 4-30.

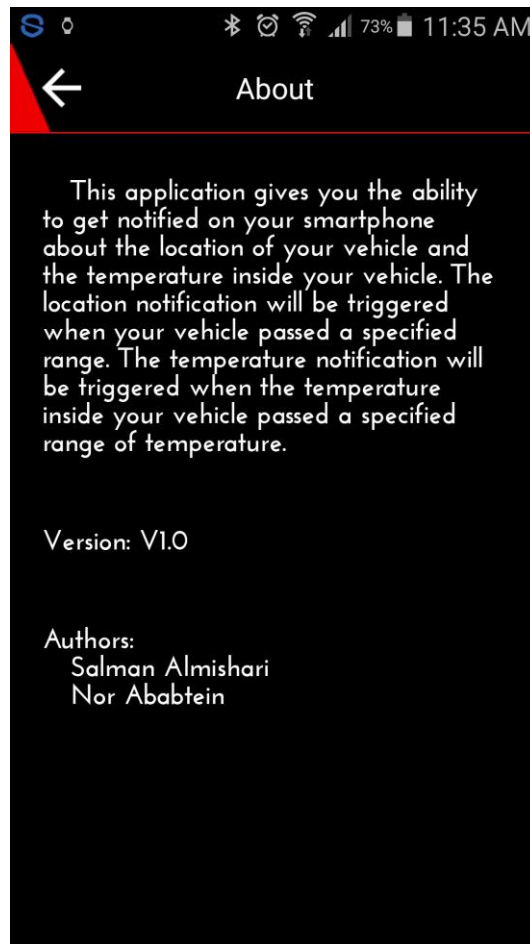


Figure 4-30: Android Application About Page

4.3.4.1 Use Cases for Developed Constraints of SVS System

- ❖ In some countries, extreme temperature or unexpected temperature changes can damage objects inside a vehicle, which can result in fires, cargo spoilage, or other critical damages. The temperature could be either too high or too low. The SVS

System will have sensors to measure the temperature continuously and notify the user when a certain temperature is reached. These sensors are connected to a framework that transmits the data to the Cloud, where data will be analyzed and a decision will be made to notify the user or not.

- ❖ Location is one of the most important aspects in vehicles. For example, some car rental companies do not allow their vehicles to go out of province or state. The SVS can constantly update the company with a vehicle's information and location. It will also send a notification to the smartphone application that has been downloaded by the customer. The SVS will send the location of the vehicle to the Cloud continuously. After the location is received, then an out of bounds check will be made. If the vehicle is out of bounds, then a notification will be sent to the company and the customer.

- ❖ This system could be used for food transportation. With food transportation, the temperature inside the cargo is critical, because some food could be spoiled if it reaches a certain temperature for a period of time. In this case, the system will continuously report the temperature inside the cargo and also will specify the location of the vehicle. The system notifies the driver by sending a notification to his/her smartphone. In addition, the system sends an email to the administration to notify them about the cargo temperature and its location.

This system could be used in many different applications such as:

- Car rental companies.
- Moving and transportation companies.
- For people with more than one vehicle.
- To prevent vehicle theft.

Chapter 5

5 Solution, Experiments and Results

This chapter discusses the recommended power reduction algorithm solution, the experiment setup, and the experiments that have been conducted in this work and their results. Section 5.1 describes the solution algorithm used to reduce the Tracking Unit's power consumption. Section 5.2 explains the experiments' setups and how the devices are connected. Section 5.3 defines some scenarios where the system could be used. Section 5.4 gives detailed results, explanations and the measurements of the ten different static experiments, and Section 5.5 presents the results of the four different dynamic experiments.

5.1 Suggested Power Saving Methodology for the Tracking Unit

The Tracking Unit is developed to upload the location, speed, and temperature of a vehicle every specified period of time. For example, the data can be uploaded to the Cloud every 10, 20, 30, 40, or 50 seconds. The normal workflow of the Tracking Unit goes through three stages:



1. The sensors sense the environment (temperature, location, speed)
2. The microprocessor records all the output from the sensors.
3. The GPRS/GPS Quadband Module uploads the output to the cloud every period of time via GSM network.

Our experiments showed that the peak power consumption occurs when uploading the data to the Cloud via the GSM network. The suggested methodology is to do some processing before uploading the data to the cloud, so that a reduced number of uploaded data to the cloud decreases the power consumption. The suggested workflow that should

be implemented on the Tracking Unit goes through five stages to reduce the power consumption, as follows:



1. Sense the environment (temperature, location, speed)
2. Record all the output from the sensors.
3. Store the output on the microprocessor's memory.
4. Compare the new recoded output to the previous one.
5. If constraints are met, upload the output to the cloud every set time period via GSM network.

In this methodology, we have added two more stages: store and compare. In the store stage, the microprocessor stores the sensors' output, that is the location and temperature, as variables in its memory. After we store the data, we enter the compare stage, which compares the location and temperature of the new record with the previous stored data. For location, the longitudes and the latitudes are compared. If they are the same, then we compare the temperatures. If the difference between the two temperatures is less than the threshold, then the data is not uploaded. Otherwise, the data is uploaded to the cloud.

An example of the three-stage message sequence inside the Tracking Unit is shown in Figure 5-1. The first stage starts after the Tracking Unit is powered on and the system is connected to the GSM network. The Arduino microprocessor sends a reading request to the sensors (GPS and Temperature) via a physical connection. Then, the sensors sense the surrounding environment and send records back to the Arduino. In the first stage, the Arduino stores the reading and waits for a specified time such as 10, 20, 30, or 50 seconds. Afterward, we enter Stage 2, where the second loop starts. The Arduino sends a reading request to the sensors and gets the records. These records are compared with the records from Stage 1; if the two are the same, the microprocessor does not send them to the GSM/GPRS, and then waits to enter Stage 3. In Stage 3, the same loop is repeated, where the Arduino sends a reading request again to the sensors and gets new records. These records are compared with the records from Stage 2; if the records are different,

they are stored and sent to the GSM/GPRS, while the microprocessor waits for another loop.

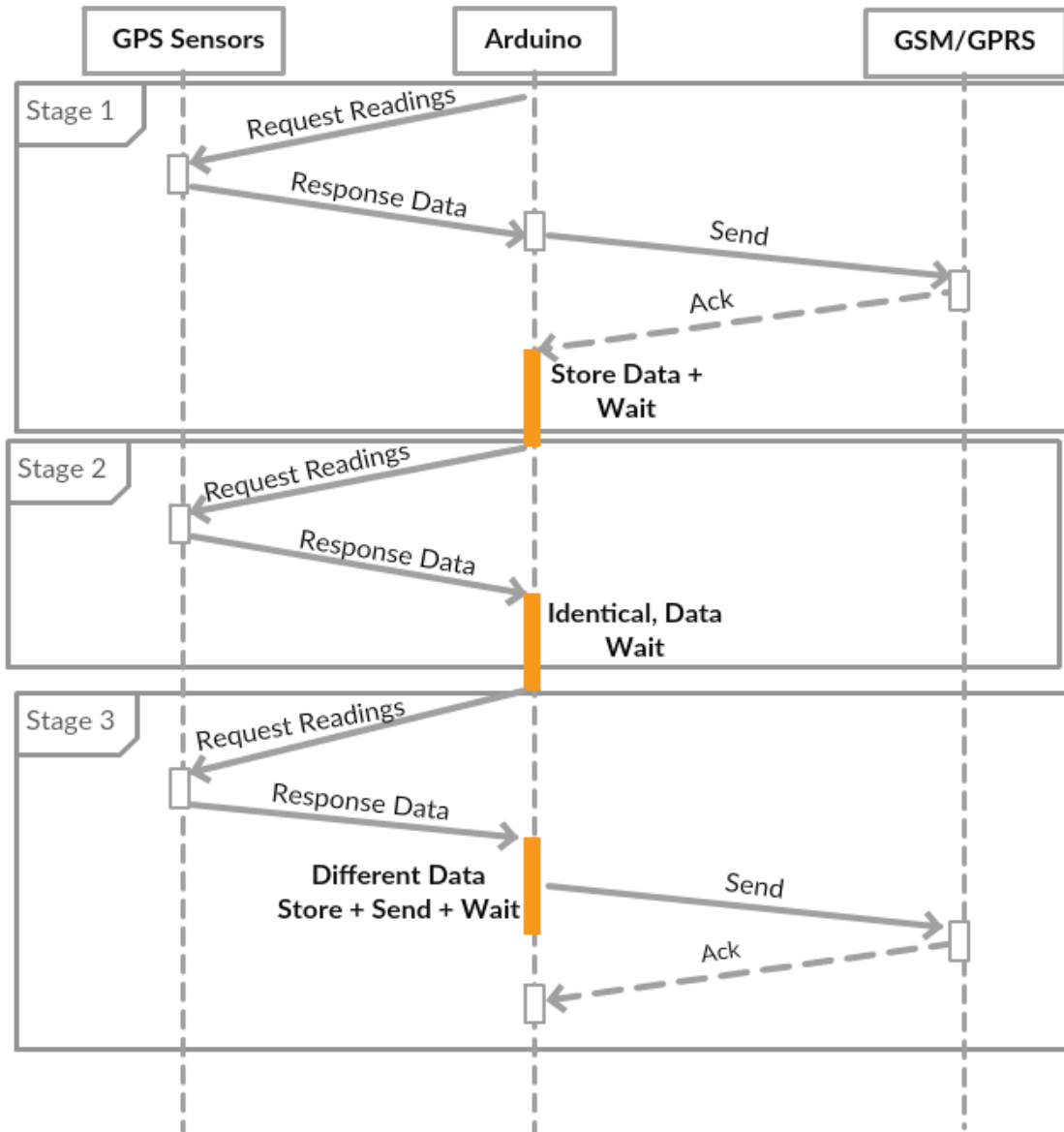


Figure 5-1: Sequence Message for Power Saving Algorithm with Different Stages

5.1.1.1 Pseudo Code After Applying the Suggested Power Saving Algorithm

Sense temperature, location, and speed

If location is identical to the previous location

If temperature is identical to the previous temperature

If difference between previous temperature and current temperature is in threshold

Don't upload the data.

Then, upload the data.

Then, upload the data.

The sensors that are connected to the Arduino board sense the temperature, location, and speed, and then the Arduino processor processes this data. First it checks the location and compares it with the previous record. If locations are not identical, it uploads the data to the cloud instantly. If they are identical, it compares the current temperature and the previous temperature. If they are not identical, it performs another check, comparing current and previous temperatures. If they are within the threshold, then it does not upload the data. If they are not different, it uploads the data. The threshold in our system was set not to exceed one degree Celsius. Thus, we ignore any decimal points between temperatures. After these processes are done, a new cycle starts, and all of these processes are repeated.

5.2 Experimental Setup

This experiment is conducted to measure the power consumption of the Tracker Unit (TU), the device that would be implanted in a vehicle to sense the environment and send the sensors' output to the cloud. The TU is designed to be portable, and its main power source is a battery. The main goal of this experiment is to try to expand the battery life of such a system. The experiment setup consists of three different components: the Tracker Unit, a laptop, and a Monsoon Power Monitor (MPM).

The Tracker Unit has the following elements: the Arduino UNO, GSM shield, GPS antenna, GSM antenna, and temperature sensor. All of these elements are described in more detail in section 4.1. The TU is used to get the location, speed, and temperature of a vehicle and send them to the cloud via the GSM network.

The laptop used in this experiment is an ASUS 14", which has the following features: a 2.16GHz Intel Pentium quad core processor, 1TB hard drive, 8GB RAM, and Windows 10. This laptop is used to monitor the Arduino's activities as well as to control, view, and store the power consumption readings of the MPM device. Power tool software with a Graphical User Interface (GUI) provided by Monsoon Solutions Inc. was installed on the laptop. This software is used to get all the readings and to monitor the power consumption of the TU.

This software GUI gives a large number of details about the measurements, as shown in Figure 5-2. The measurements that we have selected to be displayed on the graph in this GUI are the average power and current that the microprocessor uses. In addition, we have changed the graph's scale as follows:

- 1- Time (along the bottom) to be shown every 100 millisecond (ms) with a gap of 10 units.
- 2- Power on the left side with the scale gap set to 500 milliWatts (mW).
- 3- Current on the right side, with the scale gap set to of 500 milliAmpere (mA).

On the other hand, the graph in Figure 5-2 shows exactly how many samples of measurements have been captured, the total consumed energy, average power, and average voltage for the whole measured time. We set the battery size to 550 milliAmpere per hour (mAh), which is the size of a 9V battery, to get the expected battery life in hours.

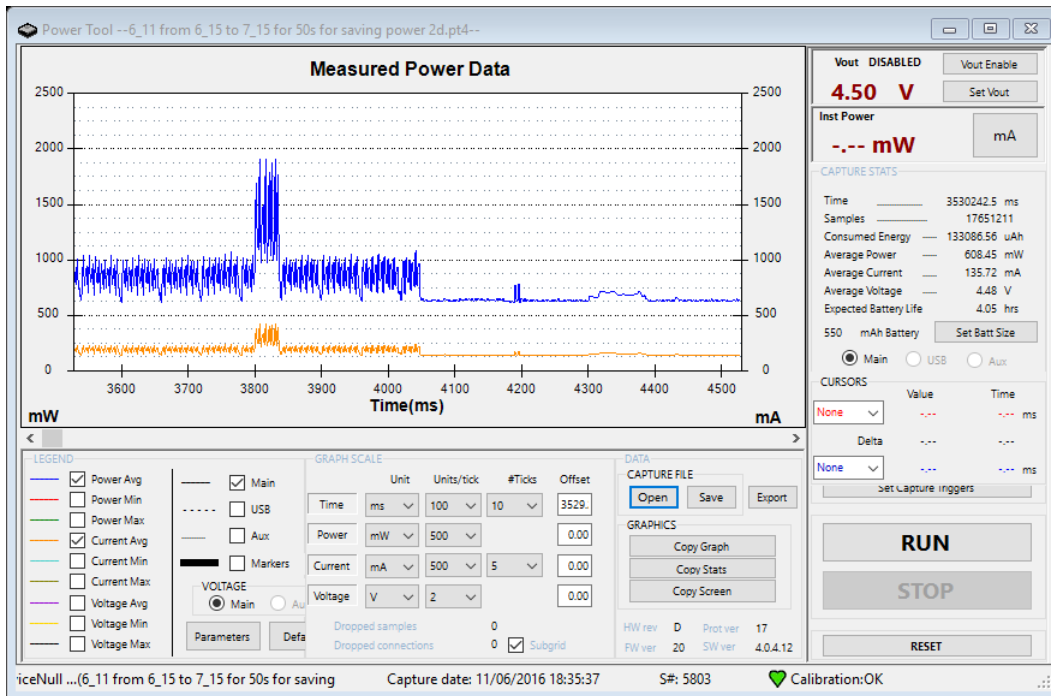


Figure 5-2: Monsoon Application GUI

The Monsoon Power Monitor (MPM) device is used to power the Tracking Unit and measure all the outcome of the power provided to the TU. This device is connected to the TU and the laptop, which has the power tool software. A detailed description of the software GUI used to manage these measurements was provided in the previous paragraph. The MPM device is connected to the laptop though a Universal Serial Bus (USB) cable to transfer the captured measurements from the MPM device to the laptop. On the other side of the MPM, there are three different channels that measure data: the Main, USB, and Auxiliary channels. These experiments use the Main channel, which has red and black terminals connected to the Tracking Unit to provide power, as shown in Figure 5-3. The red terminal is connected to the 5V pin on the Arduino board. This pin also used to power the temperature sensor. The black terminal is connected to the black cable of the jack connector.

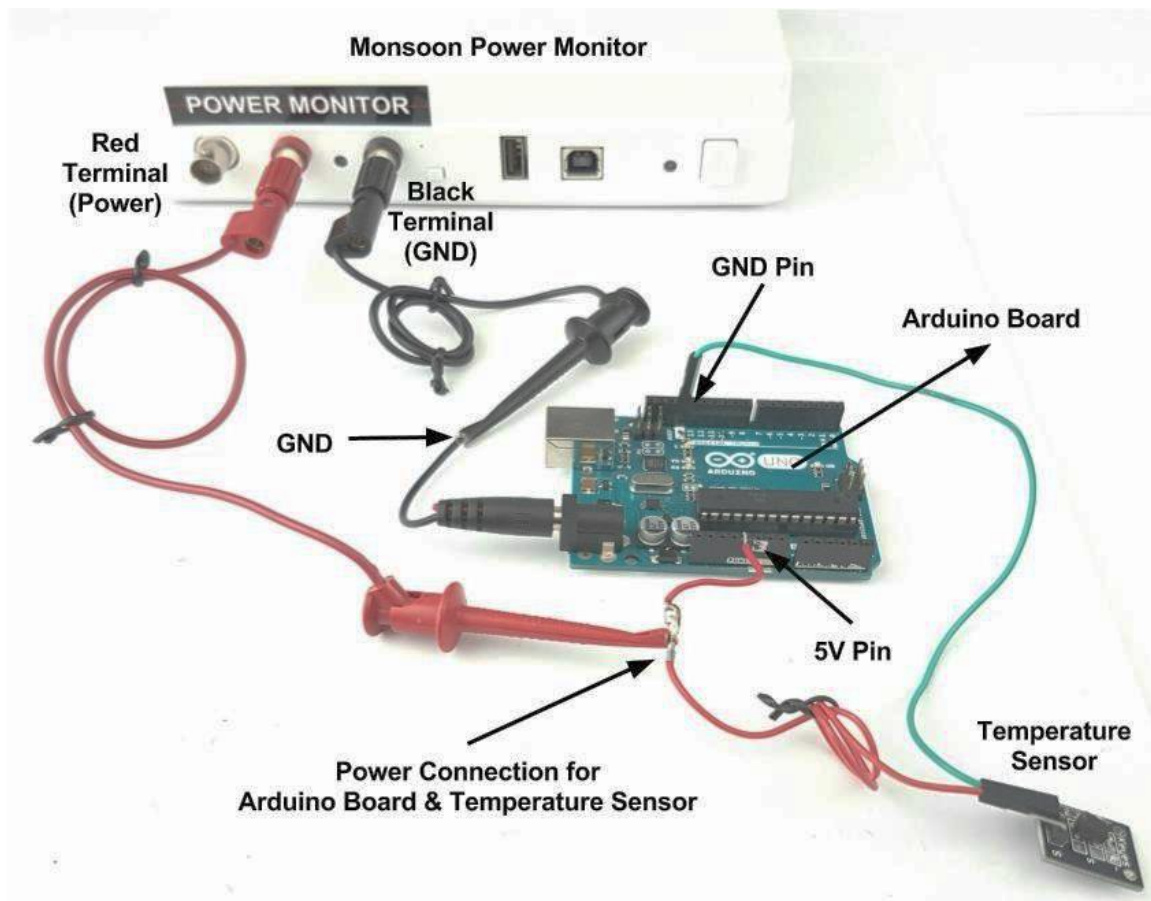


Figure 5-3: Power Connection Between Monsoon Power Monitor and Arduino Board

The Monsoon Power Monitor device gives power of only up to 4.8 volt, insufficient to power the Arduino microprocessor and the GSM/GPRS module at the same time. For that reason, at the beginning of each experiment, a USB connection between the laptop and the Arduino is used, to help power both. After the first data upload, this connection is disconnected, and the measurements start. When the experiment finishes, the measurements are stored. Then, we go through the following steps to start the next experiment:

- 1- Stop the MPM device by disabling the voltage from the MPM device's GUI.
- 2- Save the measurement data file that is generated from the MPM device on the laptop.
- 3- Take a screen shot of the readings from the MPM device's GUI.
- 4- Download the CVS file from the cloud database that has all the uploaded records.
- 5- Take a screenshot of database details.

- 6- Reset the database on the cloud.
- 7- Change the Arduino code to the required upload frequency, using the Arduino environment.
- 8- Unplug the GSM/GPRS module from the Arduino board.
- 9- Upload the new sketch to the Arduino board using the Arduino environment.
- 10- Place the GSM/GPRS module back on top of the Arduino board.
- 11- Connect the red cable to the 5V pin on the Arduino board.
- 12- Restart the MPM software.
- 13- Connect the USB cable between the laptop and the Arduino board.
- 14- Run the serial connection monitor using the Arduino environment.
- 15- Enable the voltage from MPM device's GUI.
- 16- After the first upload of data, disconnect the connection between the laptop and the Arduino board.
- 17- Click on "Run" on the MPM device's GUI to start recording the measurements.
- 18- Repeat the above steps starting from step number 1.

5.3 Experimental Scenarios

The main focus of the system that has been developed is to help managing fleets with a portable device. In addition, the SVS system, because of its portability, can be used for any other objects, persons, or assets that need to be tracked. The SVS system monitors a vehicle's location, speed, and temperature, and transmits that data to a remote location.

Different countries have different environments that may affect our belongings or food being transported. For example, countries like Saudi Arabia have extremely hot weather in the summer. According to a weather online website, in the summer, sometimes the temperature reaches 54° Celsius (129° Fahrenheit) [31]. This kind of temperature could damage objects inside a vehicle, cause fires, and may cause food poisoning. On the other hand, some countries have a very cold weather in the winter, such as Canada. According to the Canadian Geographic website, the lowest temperature recorded in Snag, the Yukon, Canada is —63°C [32]. Therefore, to accommodate these different climates, we enter some threshold variables, which can be adjusted by the administrator of the system. For our system, we have created two variables: the highest and lowest temperature allowed for each vehicle, as shown in Figure 5-4. These two variables can be set from the Edit Temperature Range Page on the website. If a vehicle goes beyond these two thresholds, then the system will notify the user on the Android application and the administrator by email.

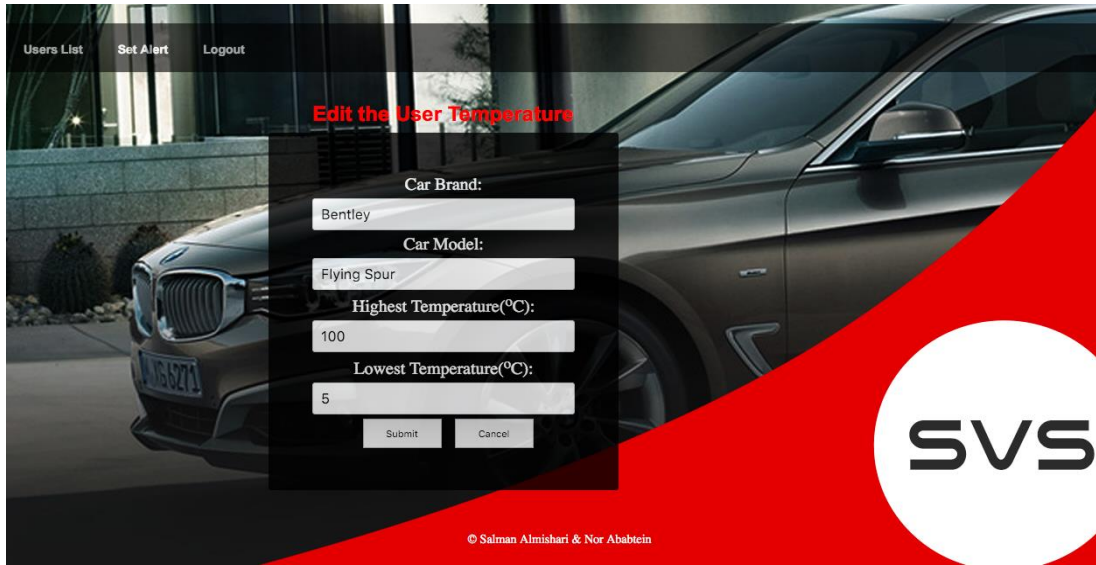


Figure 5-4: Highest and Lowest Temperature for Vehicles

In addition, the location of a vehicle is really important. For example, sometimes transportation companies do not allow their vehicle to go outside of certain zones or cities. With our system, the administrator can set a certain threshold for each vehicle. We have created a diameter range of a location specified by the administrator, as shown in Figure 5-5. To set the radius of the location diameter, which would be the starting point of the range, the icon pin in Google map is placed on the location of the radius. After that, the administrator sets the range beyond, which the vehicle should not pass. This range is calculated from the radius point. For example, the University of Waterloo has some service vehicles that must be used only in the university's area. We set the radius to be the University of Waterloo in Figure 5-5 and the range to be 500 meters. Therefore, any vehicle that uses our system and passes out of the university area violates the university's rules. If such a violation occurs, then the system will notify the vehicle's driver on the Android application and the vehicle service administrator by email.

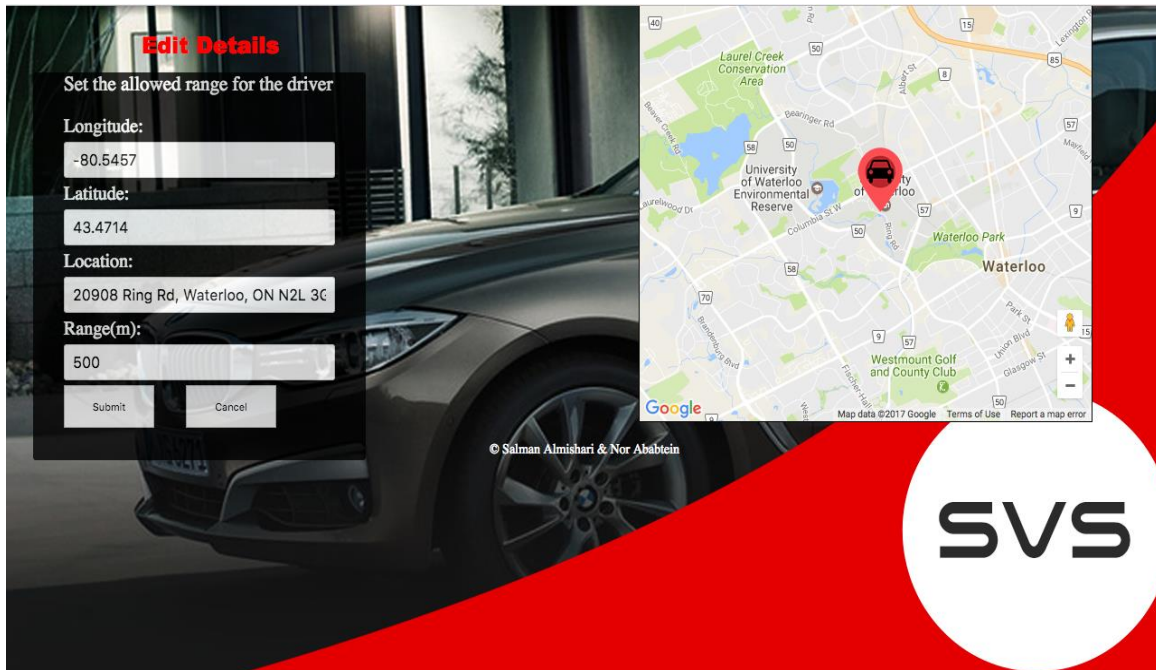


Figure 5-5: University of Waterloo as a Radius Location

5.4 Experiment for One Hour Static Location

This experiment was conducted for one hour in a fixed location for five different upload frequencies. Each upload frequency was tested for one hour. The total number of experiments is 10 in the same location. The upload frequencies that were used in this experiment are 10, 20, 30, 40, or 50 seconds. Each frequency was tested twice, before and after applying the power-saving algorithm. All the measurements are recorded and presented in this thesis.

The power saving mode will acquire the GPS coordinates and the temperature then compare them with the previous record. First, the coordinates (longitude and latitude) are examined to make sure they are not the same as the previous ones. Then, another check for the temperature is performed, to determine if both temperatures are within the same temperature degree. If both records (coordinates and temperature) are the same, then the Tracking Unit does not upload the data to the cloud. If one of them has changed, then the Tracking Unit uploads the data.

5.4.1 Detailed Results for each experiment

The detailed results readings give a summary of each experiment for the different frequencies that have been used, and also provide all the measurements recorded from the Tracking Unit (TU) side and the cloud side. From the TU, the average power, average current, average voltage, and the consumed energy are recorded. Furthermore, the expected battery life of the Duracell 9V battery (550mAh) is calculated. From the cloud side, the database size is documented. Then, the power saving and the database saving percentage are calculated. Also, the expected increase of the battery life is estimated. In the end, there will be a comparison between all of the fixed location measurements from the average power point of view. The following sections describe all the details of all the fixed location experiments.

5.4.1.1 One Hour Fixed Location Experiment for 10 Seconds

This experiment was conducted for 10 seconds upload frequency for one hour. All the measurements details are described in Table 2 below. The power saving from this proposed algorithm is 17.6% and will expand the expected battery life by 42 minutes. In addition, the database saving percentage is 82.2%.

Table 2: Fixed Experiment for 10 Seconds Results

	Regular Power	Power Saving Mode
Average Power	766 mW	631 mW
Average Current	170.8 mA	140.8 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	171.5 mAh	138.4 mAh
Expected Battery Life (550mAh)	193 minutes	235 minutes
Database Size	19.15 kb	3.4 Kb

5.4.1.2 One Hour Fixed Location Experiment for 20 Seconds

This experiment was conducted for 20 seconds upload frequency for one hour. All the detailed measurements are described in Table 3 below. The power saving from this algorithm is 11.6% and will expand the expected battery life by 27 minutes. In addition, the database saving percentage is 77%.

Table 3: Fixed Experiment for 20 Seconds Results

	Regular Power	Power Saving Mode
Average Power	700 mW	618.4 mW
Average Current	156.2 mA	137.9 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	157.1 mAh	135.2 mAh
Expected Battery Life (550mAh)	212 minutes	239 minutes
Database Size	12.18 Kb	2.8 Kb

5.4.1.3 One Hour Fixed Location Experiment for 30 Seconds

This experiment was conducted for 30 seconds upload frequency for one hour. All the detailed measurements are described in Table 4 below. The power saving from this algorithm is 10.04% and will expand the expected battery life by 24 minutes. In addition, the database saving percentage is 76.2%.

Table 4: Fixed Experiment for 30 Seconds Results

	Regular Power	Power Saving Mode
Average Power	682 mW	613.5 mW
Average Current	152.1 mA	136.8 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	149.2 mAh	155.8 mAh
Expected Battery Life (550mAh)	217 minutes	241 minutes
Database Size	10.5 Kb	2.5 Kb

5.4.1.4 One Hour Fixed Location Experiment for 40 Seconds

This experiment was conducted for 40 seconds upload frequency for one hour. All the measurement details are described in Table 5 below. The power saving from this algorithm is 8.88% and will expand the expected battery life by 17 minutes. In addition, the database saving percentage is 62.2%.

Table 5: Fixed Experiment for 40 Seconds Results

	Regular Power	Power Saving Mode
Average Power	668 mW	622 mW
Average Current	149 mA	138.8 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	146 mAh	133.6 mAh
Expected Battery Life (550mAh)	221 minutes	238 minutes
Database Size	6.35 Kb	2.4 Kb

5.4.1.5 One Hour Fixed Location Experiment for 50 Seconds

This experiment was conducted for 50 seconds upload frequency for one hour. All the measurement details are described in Table 6 below. The power saving from this algorithm is 7.74% and will expand the expected battery life by 19 minutes. In addition, the database saving percentage is 59.8%.

Table 6: Fixed Experiment for 50 Seconds Results

	Regular Power	Power Saving Mode
Average Power	659 mW	608 mW
Average Current	147 mA	135.7 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	143.6 mAh	133.1 mAh
Expected Battery Life (550mAh)	224 minutes	243 minutes
Database Size	5.22 Kb	2.1 Kb

5.4.1.6 Conclusion of these 10 Fixed Location Experiments

We conclude that if we use a higher frequency rate such as 10 seconds of uploading data to the cloud we save more power. As shown in the graph (Figure 5-6), when the upload frequency is set to 50 seconds, the percentage saved is less than 8 percent, while for 10 seconds frequency, it is more than 17 percent. However, we found that when we change the frequency to 20s instead of 10s, the power saving percentage drops dramatically. On the other hand, the variation between 20, 30, 40, and 50 seconds is steadier than the variation between 10s and 20s.

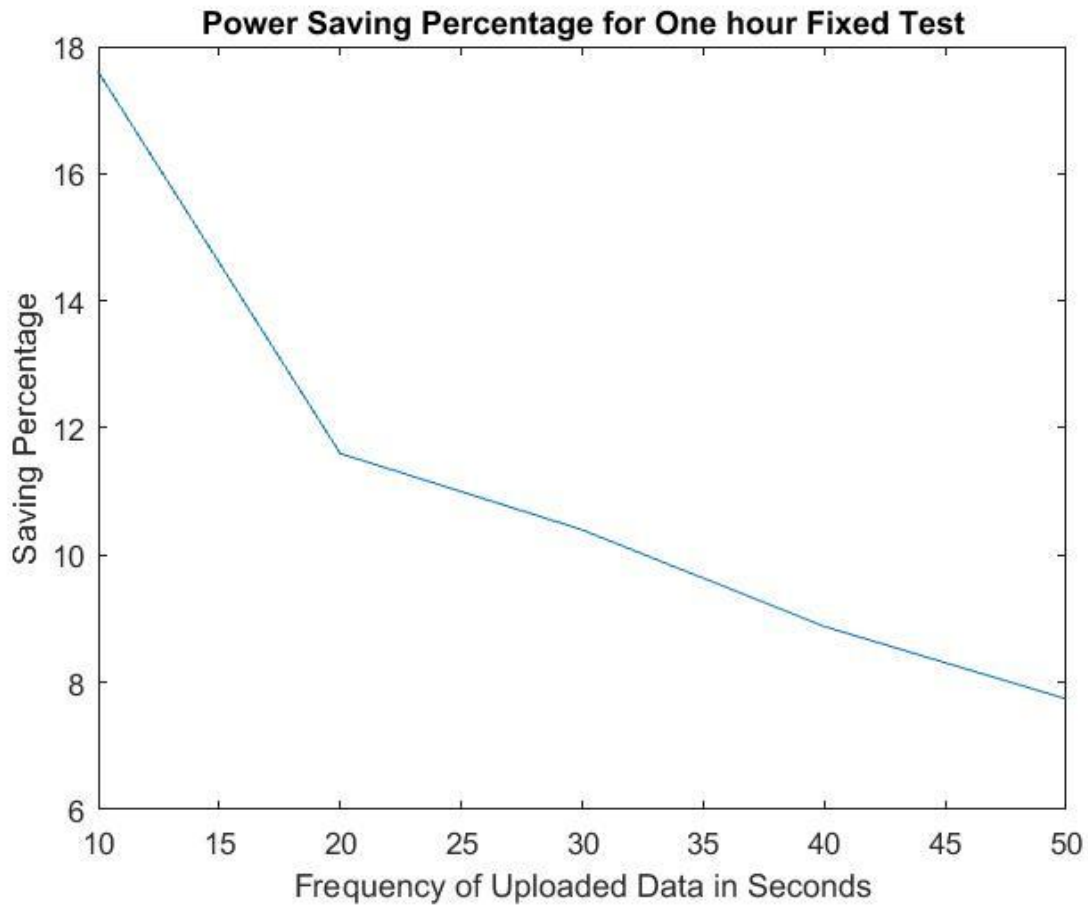


Figure 5-6: Power Saving Percentage for all of the 10 Experiment

The average power comparison between the regular and saving mode for the TU is shown in Figure 5-7 and in Table 7. The figure shows that the gap between the two lines at 10 seconds is bigger than at 50 seconds upload frequency. This is due to the amount of data uploaded to the database. The less the upload frequency, the higher the power consumption. One of the challenges that we faced in these experiments is the temperature change, because we cannot control the temperature inside the vehicle, especially when doors or windows are opened. For instance, 40 seconds upload frequency has less power saving than 50 seconds.

Table 7: Average Power Comparison Between Normal and Power Saving Mode

	Regular Power	Power Saving Mode
10 Seconds	766 mW	631 mW
20 Seconds	700 mW	618.4 mW
30 Seconds	682 mW	613.5 mW
40 Seconds	668 mW	622 mW
50 Seconds	659 mW	608 mW

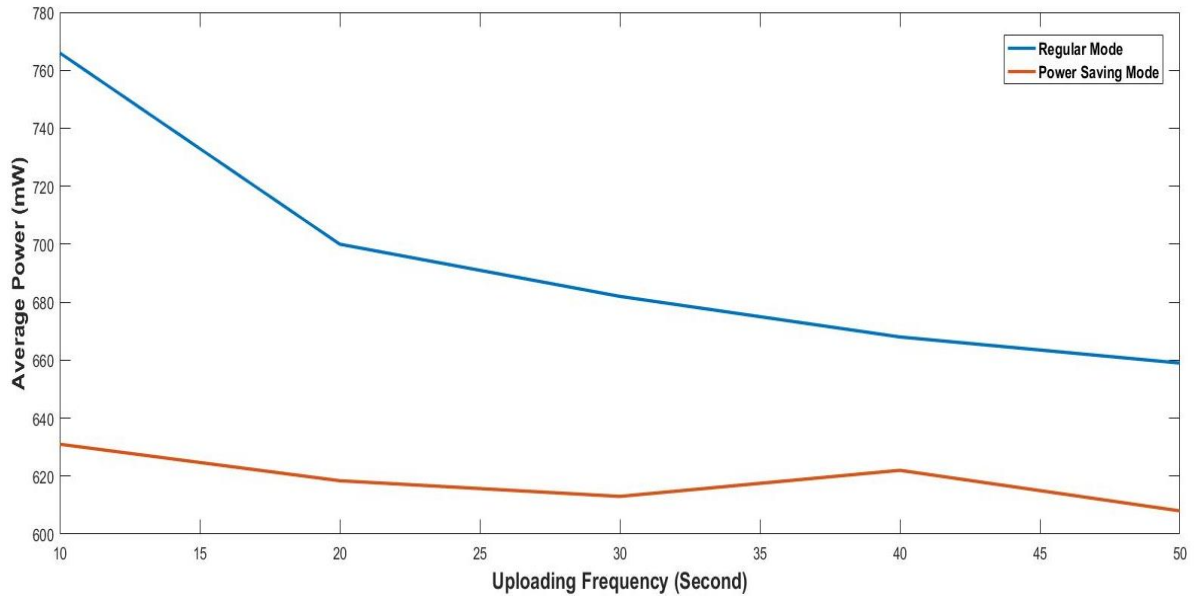


Figure 5-7: Average Power Comparison Between Normal and Power Saving Mode for Tracking Unit

5.5 Experiment for One Hour Static Location Without Temperature Sensor

This experiment was similar to the previous static experiments in Section 5.4; the only difference was that the temperature sensor was removed from the Tracking Unit (TU) and all the code was modified accordingly. The experiment was conducted for one hour in a fixed location for the same five different upload frequencies. Hence, the total number of experiments is 10 in the same location. Each frequency was tested twice, before and after disconnecting the temperature sensor. All the measurements were recorded and are presented next.

5.5.1.1 One Hour Fixed Location Experiment for 10 Seconds Without Temperature Sensor

This experiment was conducted for 10 seconds upload frequency for one hour. All the measurement details are described in Table 8 below. The average power for the TU after disconnecting the temperature sensor is 763 milliWatts, a saving of 3 milliWatts of power. Therefore, the reduction of the average power consumption for the temperature sensor represents .0039% of the total average power consumption of the TU. In this case, we can assume that the power consumption of this temperature sensor is 3 milliWatts.

Table 8: Comparison of TU Measurements for 10 Seconds Upload Frequency Results

	Measurements With Temperature and GPS sensor	Measurements Without Temperature sensor
Average Power	766 mW	763 mW
Average Current	170.8 mA	170.1 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	171.5 mAh	170.2 mAh
Expected Battery Life (550mAh)	193 minutes	194 minutes
Database Size	19.15 Kb	18.02 Kb

5.5.1.2 One Hour Fixed Location Experiment for 20 Seconds Without Temperature Sensor

This experiment was conducted for 20 seconds upload frequency for one hour. All the measurement details are described in Table 9 below. The average power for the TU after disconnecting the temperature sensor is 681 milliWatts, a saving of 19 milliWatts of power. Therefore, the reduction of the average power consumption for the temperature sensor represents .0271% of the total average power consumption of the TU. In this case, we can assume that the power consumption of the temperature sensor using 20-second upload frequency is 19 milliWatts.

Table 9: Comparison of TU Measurements for 20 Seconds Upload Frequency Results

	Measurements With Temperature and GPS sensor	Measurements Without Temperature sensor
Average Power	700 mW	681 mW
Average Current	156.2 mA	156.2 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	157.1 mAh	153.3 mAh
Expected Battery Life (550mAh)	212 minutes	211 minutes
Database Size	12.18 Kb	11.7 Kb

5.5.1.3 One Hour Fixed Location Experiment for 30 Seconds Without Temperature Sensor

This experiment was conducted for 30 seconds upload frequency for one hour. All the measurement details are described in Table 10 below. The average power for the TU after disconnecting the temperature sensor is 672 milliWatts, a saving of 10 milliWatts of power. Therefore, the reduction of the average power consumption for the temperature sensor represents .0147% of the total average power consumption of the TU. In this case, we can assume that the power consumption of the temperature sensor using 30-second upload frequency is 10 milliWatts.

Table 10: Comparison of TU Measurements for 30 Seconds Upload Frequency Results

	Measurements With Temperature and GPS sensor	Measurements Without Temperature sensor
Average Power	682 mW	672 mW
Average Current	152.1 mA	149.9 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	149.2 mAh	149.4 mAh
Expected Battery Life (550mAh)	217 minutes	220 minutes
Database Size	10.5 Kb	6.9 Kb

5.5.1.4 One Hour Fixed Location Experiment for 40 Seconds Without Temperature Sensor

This experiment was conducted for 40 seconds upload frequency for one hour. All the measurement details are described in Table 11 below. The average power for the TU after disconnecting the temperature sensor is 664 milliWatts, a saving of 4 milliWatts of power. Therefore, the reduction of the average power consumption for the temperature sensor represents .0060% of the total average power consumption of the TU. In this case, we can assume that the power consumption of the temperature sensor using 40-second upload frequency is 4 milliWatts.

Table 11: Comparison of TU Measurements for 40 Seconds Upload Frequency Results

	Measurements With Temperature and GPS sensor	Measurements Without Temperature sensor
Average Power	668 mW	664 mW
Average Current	149 mA	148 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	146 mAh	146 mAh
Expected Battery Life (550mAh)	221 minutes	223 minutes
Database Size	6.35 Kb	6.1 Kb

5.5.1.5 One Hour Fixed Location Experiment for 50 Seconds Without Temperature Sensor

This experiment was conducted for 50 seconds upload frequency for one hour. All the measurement details are described in Table 12 below. The average power for the TU after disconnecting the temperature sensor is 656 milliWatts, a saving of 3 milliWatts of power. Therefore, the reduction of the average power consumption for the temperature sensor represents .0045% of the total average power consumption of the TU. In this case, we can assume that the power consumption of the temperature sensor using 50-second upload frequency is 3 milliWatts.

Table 12: Comparison of TU Measurements for 40 Seconds Upload Frequency Results

	Measurements With Temperature and GPS sensor	Measurements Without Temperature sensor
Average Power	659 mW	656 mW
Average Current	147 mA	146 mA
Average Voltage	4.48 V	4.48 V
Consumed Energy	143.6 mAh	146.5 mAh
Expected Battery Life (550mAh)	224 minutes	226 minutes
Database Size	5.22 Kb	5.02 Kb

5.5.1.6 Conclusions from comparing Average Power consumption of TU with and without Temperature Sensor

We conclude that the difference between the average power consumption of the Tracking Unit (TU) with and without the temperature sensor changes as the upload frequencies change. As shown in the graph (Figure 5-8), when the upload frequency is set to 50 seconds, the difference is only 3 milliWatts, while for 20 seconds frequency it is 19 milliWatts. Therefore, the temperature sensor's power consumption varies with different upload frequencies, which makes it difficult to predict the sensor's power consumption. However, the variation of average power consumption differs in all cases, but still the TU consumes less power without the temperature sensor, as shown in Figure 5-8. Hence, adding additional sensors to the TU consumes more power with respect to the uploading frequency.

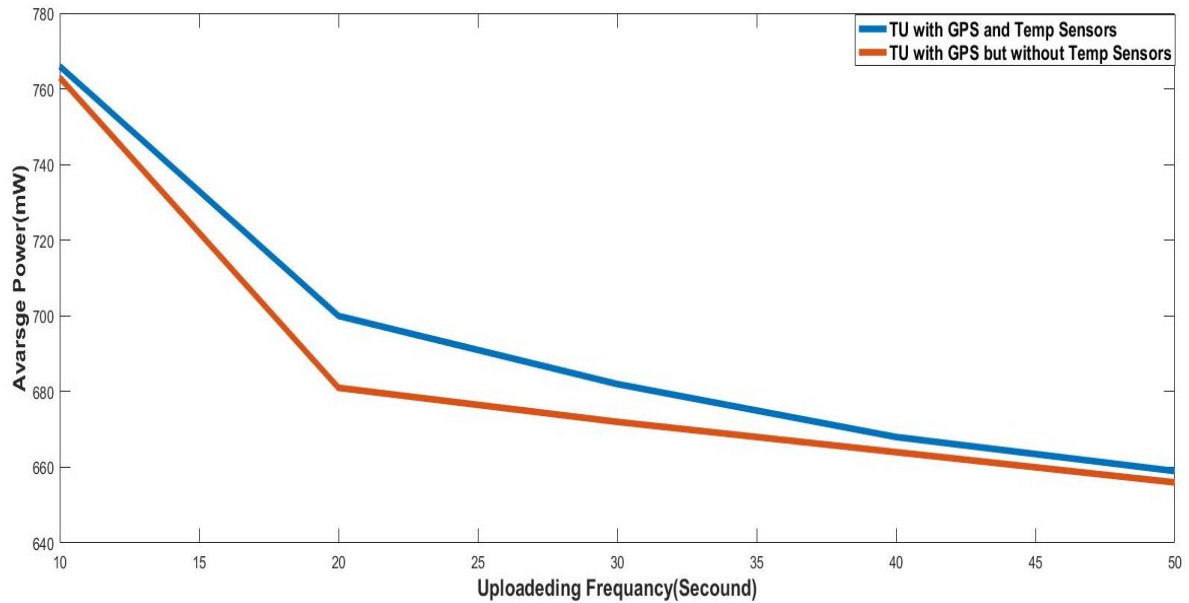


Figure 5-8: Comparison of Average TU Power, with and without Temperature Sensor

5.6 Dynamic Experiment for One Hour

The experiment was conducted to measure the power consumed for one hour of traveling to diverse locations for two different upload frequencies. Each upload frequency was tested for one hour. The upload frequencies that were used in this experiment were 10 and 40 seconds. Each frequency was tested twice, before and after applying the power saving algorithm.

The power saving mode will acquire the GPS coordinates and the temperature then compare them with the previous record. First, the coordinates (longitude and latitude) are examined to make sure they are not the same as the previous ones. Then, another check for the temperature is performed, to determine if both temperatures are within the same temperature degree (e.g., between 1 and 2° C). If both records (coordinates and temperature) are the same, then no data are uploaded to the cloud. If one of them has changed, then data is uploaded.

The experiment was designed to cover a real-life situation in which a vehicle travels from one location to another and waits for a period of time; for example, the vehicle moves from point A to point B and waits a period of time. Then, the vehicle travels from point B to point C and waits for 10 minutes. After that, the vehicle moves to point D and waits again as shown in Figure 5-9. This route is intended to measure the power consumption so it can cover both circumstances: vehicles stopping for a while then moving to other locations.

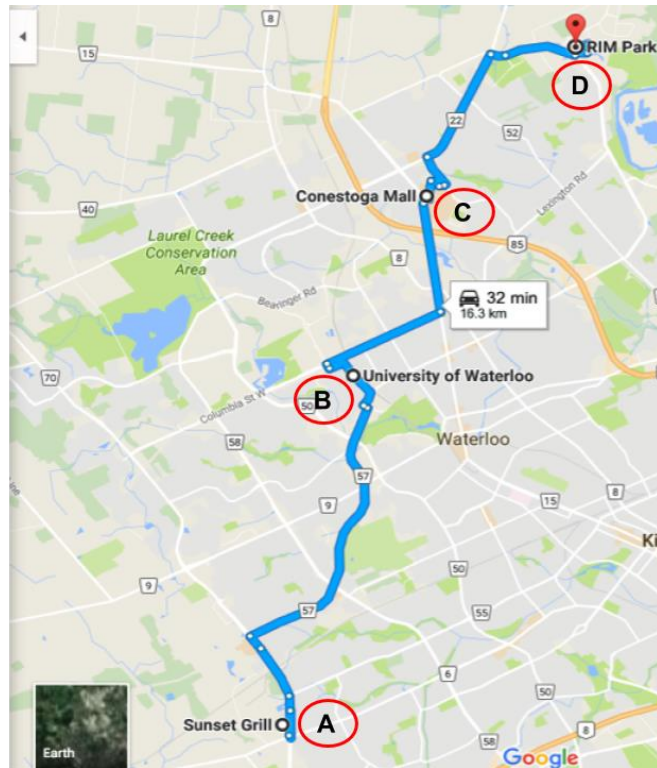


Figure 5-9: Planed Dynamic Experiment Route

5.6.1 Real-Live Dynamic Experiment Scenario

Consider a food transport company called Sam’s Delivery that delivers cooked food (usually pizza) to customers. The company picks up more than one order from several restaurants and delivers them to customers. In this experiment, we used a restaurant called the Sunset Grill, then distributed the food to three different customers in different locations.

These three customers placed an order with the restaurant. Then, Sam’s Delivery was notified with locations of the deliveries. The details of the delivery route governed where the power measurements took place, as shown in Figure 5-10. The starting place was the Sunset Grill and the finale place was Rim Park. The restaurant’s pick-up location was 235 Ira Needles Blvd., Kitchener, which is where the measurements started.

The first delivery was to the EIT building at the University of Waterloo. The drive was 12 minutes. Then, the delivery agent stopped for 10 minutes to hand over the food and get payment. After that, the agent delivered the second order to Conestoga Mall, a drive of 10 minutes, followed by a 10-minute pause. After that, the agent delivered the third order to the Rim Park, taking 13 minutes, followed by a stop of 5 minutes to hand over the food and get payment.

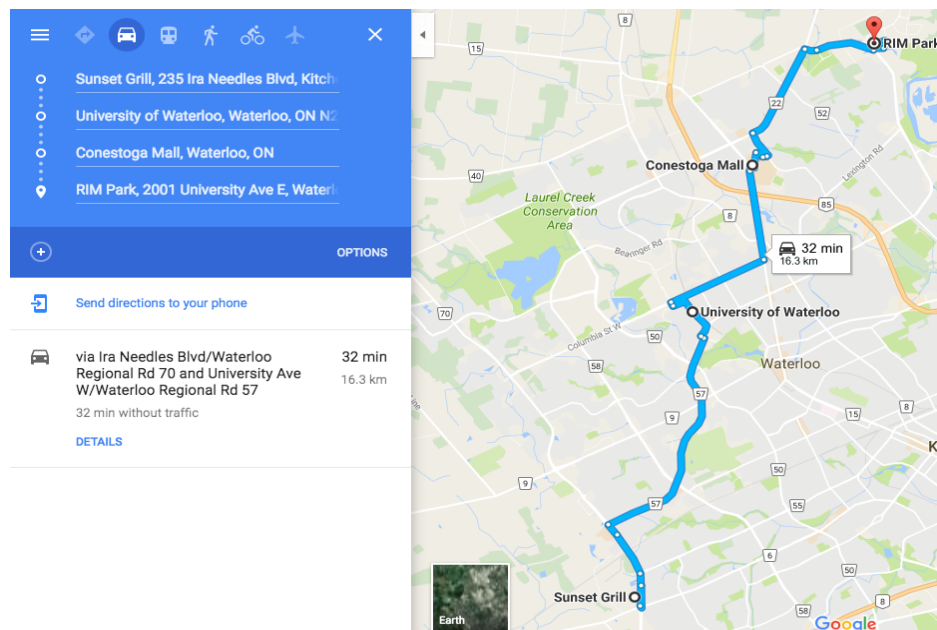


Figure 5-10: Real Live Dynamic Experiment Route

In summary, there were three orders to be delivered to three different locations. The power measurements started with the restaurant and ended at the final destination, Rim Park. Wait times for the first and second deliveries were 10 minutes, and the third delivery took 5 minutes. The total driving time was 35 minutes for 16.8 kilometers, and the time that the vehicle was not moving was 25 minutes. In total, four different experiments were examined before and after implementing the power saving algorithm. The results of these experiments are as follows:

5.6.1.1 One-Hour Dynamic Experiment for 10 Seconds

This experiment was conducted for a 10 seconds upload frequency for one hour. All the measurement details are described in Table 13 below. The power saving from this algorithm is 10.6% and will expand the expected battery life for 24 minutes. The measurements in the normal mode are based on 18,507,843 samples taken from the Monsoon power measurement tool, as shown in Figure 5-11 below. Moreover, for the power saving mode, the measurements are based on 16,997,078 samples, as shown in Figure 5-12.

Table 13: Dynamic Experiment for 10 Seconds Results

	Normal Mode	Power Saving Mode
Average Power	729 mW	652 mW
Average Current	162.5 mA	145.5 mA
Average Voltage	4.49 V	4.48 V
Consumed Energy	141.8 mAh	137.4 mAh
Expected Battery Life (550mAh)	203 minutes	227 minutes

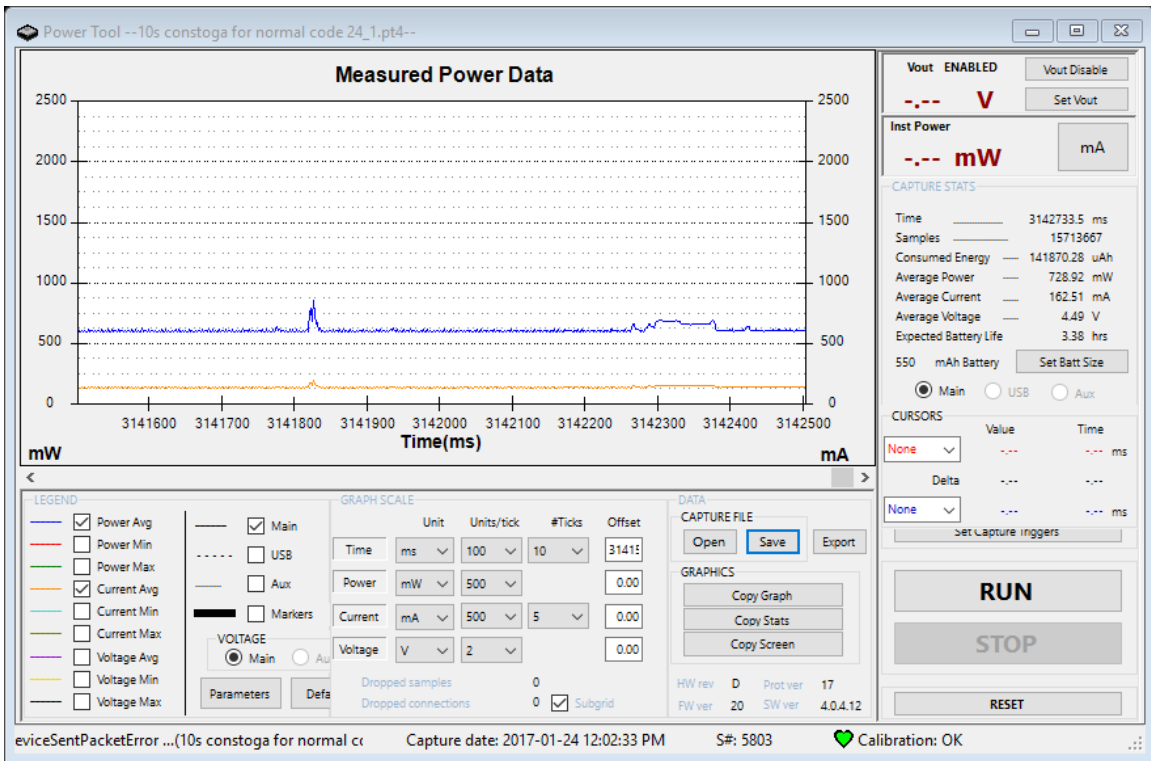


Figure 5-11: Power Measurement for 10 Seconds with Normal Mode

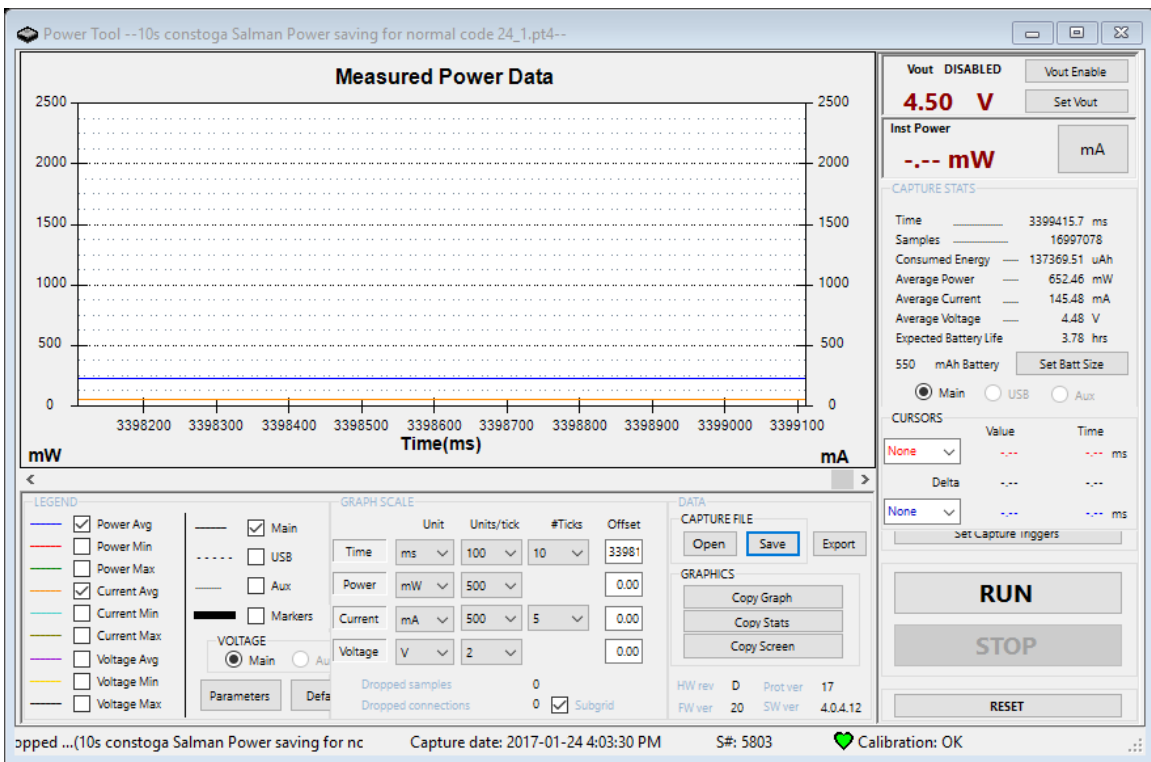


Figure 5-12: Power Measurement for 10 Seconds with Power Saving Mode

5.6.1.2 One-Hour Dynamic Experiment for 40 Seconds

This experiment was conducted for a 40 seconds upload frequency for one hour. All the measurement details are described in Table 14 below. The power saving from this algorithm is .003% and will expand the expected battery life for 10 minutes. The measurements in the normal mode are based on 17,961,872 samples taken from the Monsoon power measurement tool, as shown in Figure 5-13. Moreover, for the power saving mode, the measurements are based on 18,098,061 samples, as shown in Figure 5-14.

Table 14: Dynamic Experiment for 40 Seconds Results

	Normal Mode	Power Saving Mode
Average Power	628 mW	626 mW
Average Current	140 mA	140 mA
Average Voltage	4.49 V	4.48 V
Consumed Energy	139.6 mAh	140.4 mAh
Expected Battery Life (550mAh)	227 minutes	237 minutes

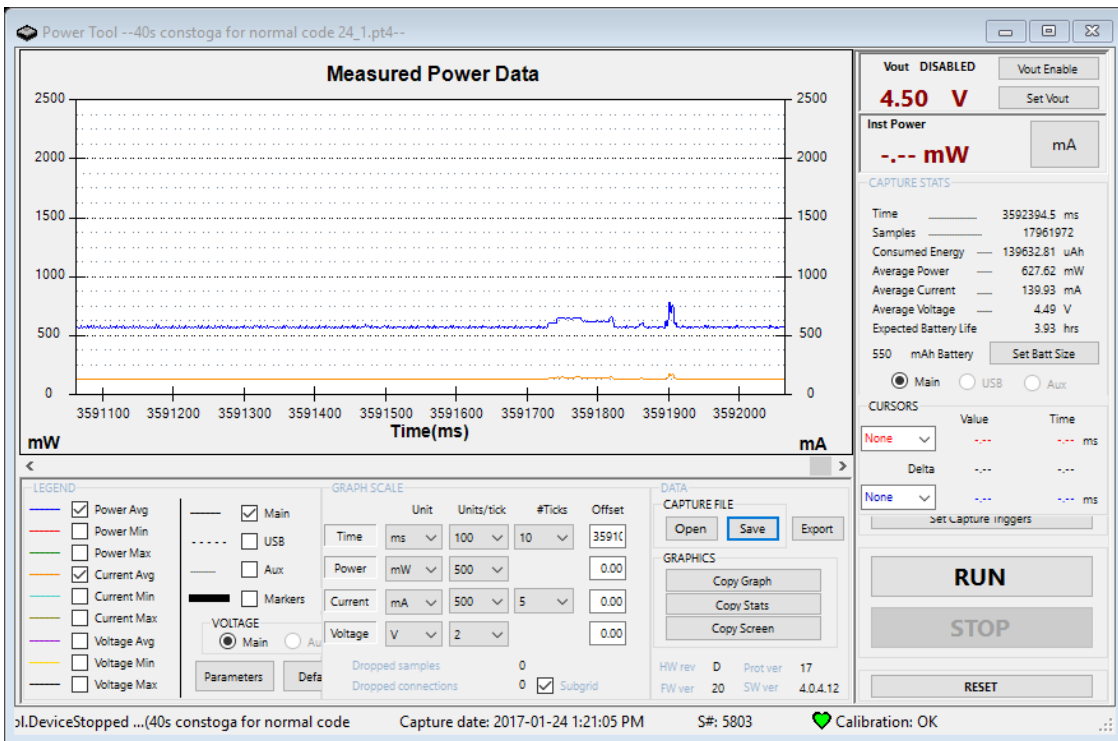


Figure 5-13: Power Measurement for 40 Seconds with Normal Mode

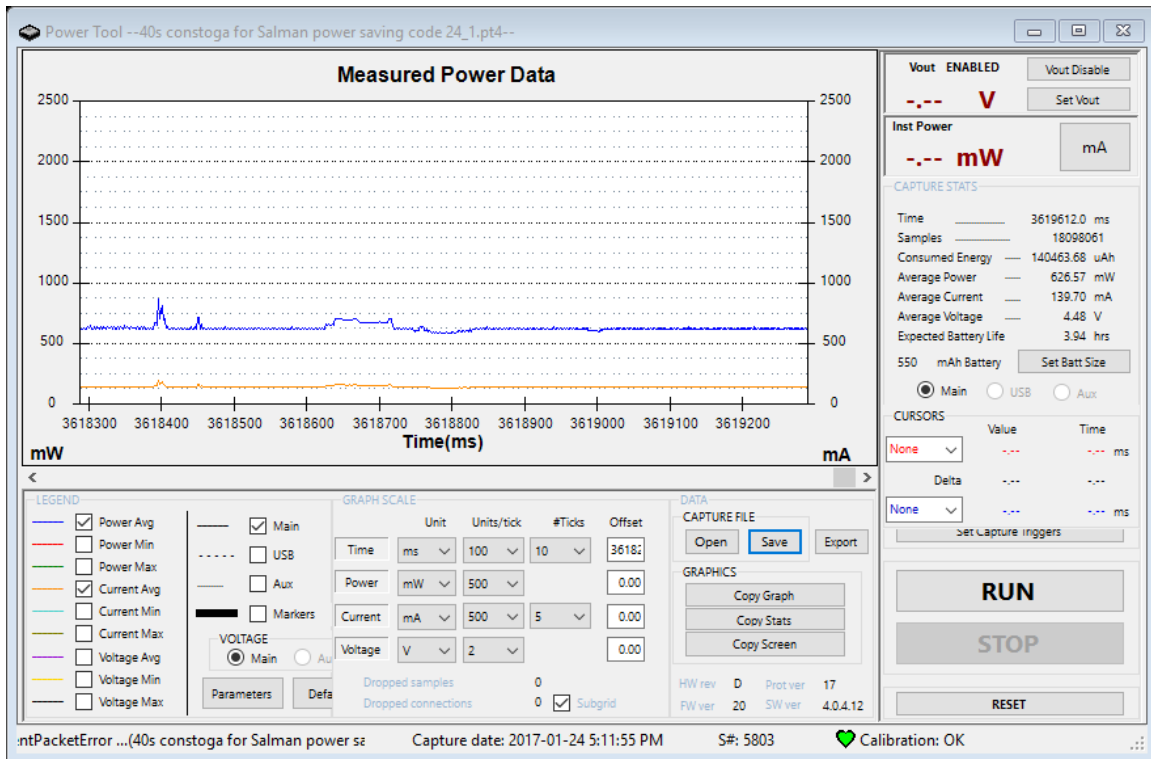


Figure 5-14: Power Measurement for 40 Seconds with Power Saving Mode

From the previous two dynamic experiments, we can conclude that the power saving mode will not be as beneficial when the frequency of data uploading increases. Hence, the less the frequency of data uploaded to the cloud, the more power saving.

5.6.1.3 Conclusion of the Four Dynamic Location Experiments

To conclude, the graph in Figure 5-15 compares power-saving and normal mode for all four dynamic experiments. It shows the average power in milliWatts (mW), the two modes, and the two frequencies (10 and 40 seconds), used in these experiments. From the graph, we can conclude that there is a significant difference in power consumption between the two different frequencies. The difference in the normal mode between the two frequencies is more than the difference in the power saving mode. This difference clearly explains that when we use a lower upload frequency, we save more power. Moreover, the power saving for 10 seconds is larger than the power saving for 40 seconds, because the expected upload to the cloud for 10s is 360 records, but for 40s is

only 90 records. Therefore, we conserve more power when we use a low upload frequency such as 10 seconds.

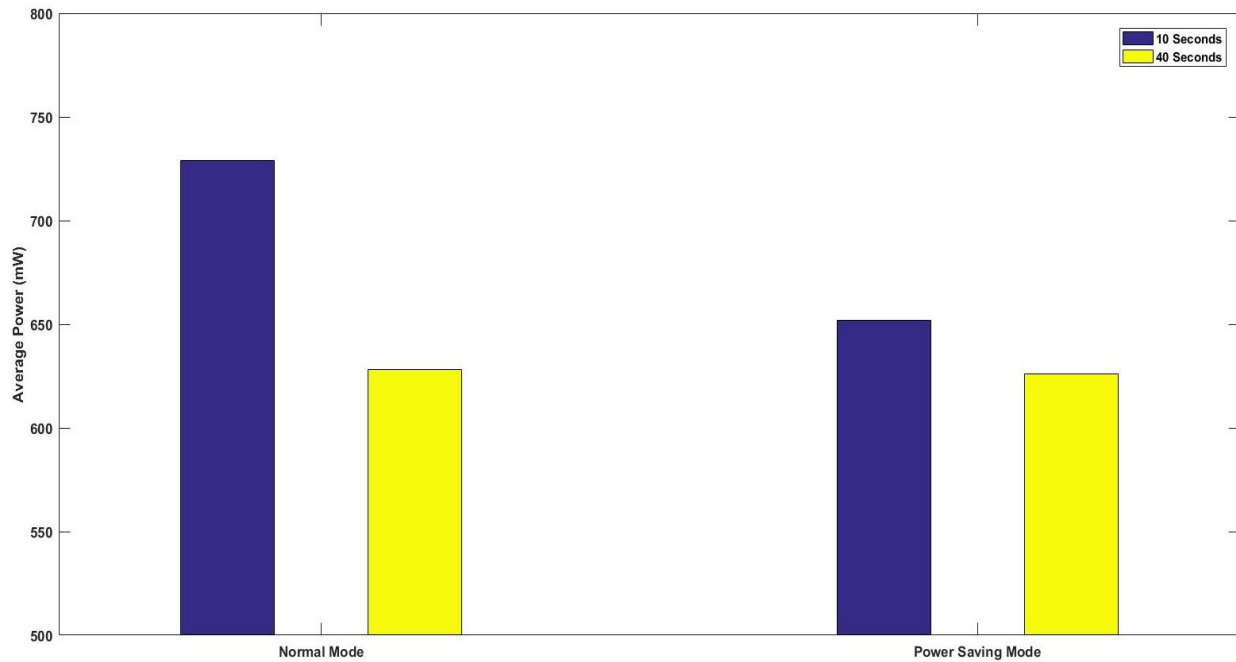


Figure 5-15: Comparison Between Normal Mode and Power Saving Mode

Chapter 6

6 Challenges, Conclusion, and Future Work

6.1 Challenges

The Smart Vehicle System (SVS) is a fully functional system with three main parts: the Tracking Unit, Web Panel, and database. Each part has its own challenges and limitations, which we have tried to overcome. We go through these changes in detail for each part of the system in the following paragraphs. Other challenges were measuring the power consumption and finding the right power-measurement device.

At the beginning of the research, our idea was to build a useable IoT system and then try to improve it. We had many ideas such as smart homes, but needed to find the right system to investigate in the limited time that we had available. Therefore, we decided to work on vehicle tracking systems because they are one of the most critical

components of self-drive cars, which are one of today's greatest trends. The Tracking Unit is a main segment of the system.

The Tracking Unit was the most challenging part of the system because there are many ways to build such a unit. Choosing a microprocessor that would be the main brain of the system was interesting. First, tried to work with the Raspberry Pi microprocessor, but unfortunately, we found that the Arduino worked better for our purposes. Then, we worked on the communication between the Arduino and the sensors. After that we looked at how to transfer the data to the cloud. Three main sensors were needed: the GPS, GSM, and Temperature. Many kinds of temperature sensors can work with the Arduino, some analog and some digital. We chose the digital one because of its accuracy. On the other hand, we had some challenges connecting the GPS antenna to the satellites to fetch location data and calculate speed. First, we used short antennas, shown in Figure 6-1, that made it hard to connect to the satellites because of a lack of line sight. We had to change these antennas to more accurate, waterproof, and longer cables that gave the GPS the ability to connect to satellites from inside a vehicle.

These antennas are connected to the GSM/GPS Module (SIM 908), which has its own power source. We faced many problems trying to change it to get its power from the Arduino board. After making sure this Tracking Unit was working, we started building the cloud side.

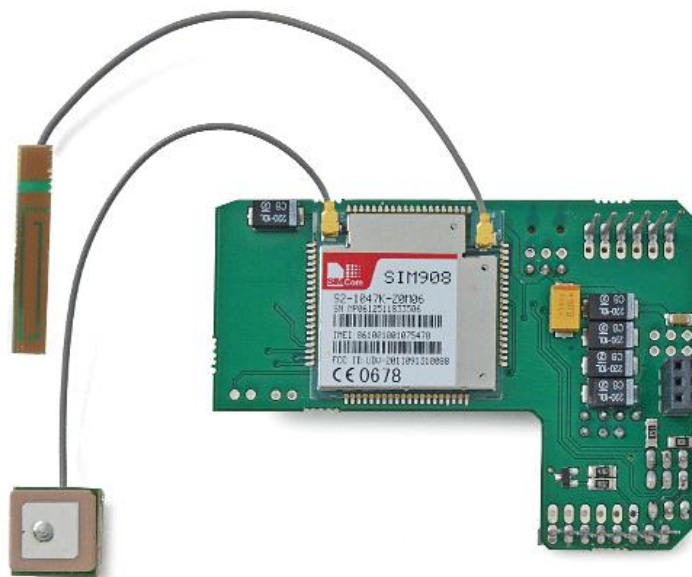


Figure 6-1: SIM 908 (GPS & GSM) Short Antennas

The cloud has two parts: the web panel and the database. Setting up the design of this web panel was not easy because we tried to make it as simple as possible. The most challenging obstacles that we faced were storing the received data, embedding Google Maps, setting up the boundaries of location and temperature, and linking the smart phone with the user's information. We used HTTP protocol to send data from the Tracking Unit. Then we read these data from the URL using PHP code, and stored them on the relational database using MySQL. To set up boundaries for the location, we used embedded Google Maps. Every time a location is inserted into the database, a boundary check is performed. For the temperature boundaries, we used PHP code to compare the temperature inserted with the maximum and the minimum temperatures already set by the administrator.

For the Android application, fetching and viewing the data coming from the database was challenging. We used embedded Google Maps to show the location of the vehicle and a scale for the temperature. For the response from the web panel, we used JSON, which includes a device token to push notifications in the driver's device. Each Android device has its own token, which is recoded onto the database and added to each driver's information.

On the experimental side, we faced many barriers, including situations that we could not control, for instance, the time it took for each experiment because of traffic and accidents. Another barrier was the changes of temperature when we examined different upload frequencies and compared them. Additionally, some areas did not have cell coverage or there was a very low signal, so we had to consider these circumstances in our experiments. We also had to make changes in how we measured the power consumption, because the Monsoon Power Monitor (MPM) provides a maximum of 4.9V and to powering up the Tracking Unit required at least 6V. Therefore, we used a USB cable connected to the laptop to support the MPM and power up the Tracking Unit. Then, after the first data upload, we disconnected the USB cable to give us more accurate readings.

6.2 Conclusion

The main goal of this research was to develop a vehicle tracking system and provide an innovative approach to the energy consumption of the Tracking Unit functionality. The Smart Vehicle System (SVS) has been presented, and tracks, records, analyzes, and notifies users of certain events. The idea of this system is to help manage fleets and prevent certain hazards.

There are three main parts to the SVS: the Tracking Unit, Cloud, and Android application. The Tracking Unit is a device that is placed inside a vehicle, and consists of an Arduino UNO microcontroller, temperature sensor, GPRS/GPS Quadband module, 9V battery, GPS antenna, and GSM/GPRS antenna. This Tracking Unit senses the vehicle's location, speed, and temperature, and then uploads them to the cloud via GSM networks. The cloud includes a web panel, web server, and database. The web panel and the webserver are used to manage and view the users and data recorded by the Tracking Unit. The main function of the database is to store these records for further processing. A detailed description of all the parts and the communication that are used in this system has been presented in Chapter 4.

The power consumption of the Tracking System is essential, which led us to propose and design an algorithm to improve the power consumption by modifying the Tracking Unit's software. Developers should consider the power consumption of such devices at the development stage. The algorithm was designed and implemented to add two more processing stages to the Tracking Unit processes, before uploading the data, to conserve power: the store and compare stages. In the store stage, the microprocessor stores the location and temperature in a variable on its memory. After we store the data, then we enter the compare stage. This stage compares the location and temperature of the new record with previously stored data. For the location, the longitude and the latitude are compared. If they are the same, then we compare the temperature. If the difference between the two temperatures is less than the threshold, then the data will not be uploaded. Otherwise, the data will be uploaded to the cloud.

Our experiments before and after applying the algorithm have been discussed, including all the setup and results of each. We have performed 14 different experiments, four of them dynamic and ten fixed location. These experiments prove that the suggested algorithm can make a difference, especially when the data uploading needs to be more frequent. As a result, we have succeeded in conserving up to 17% of the power consumed by the Tracking Unit.

6.3 Future Work

Several different alterations, investigations, and experiments have been left to the future due to the limited time that we have. Experiments with real time data are usually time consuming, and sometimes it requires days to prepare and perform a single one. Future work could involve deeper analysis of precise mechanisms, different protocols to try new methods, or different hardware to test performance. There are some different ideas and hardware that we would have liked to try during the description and development of our Smart Vehicle System in Chapter 4. This thesis has been mostly focused on designing an IoT system, minimizing power consumption, and tracking some aspects of a vehicle such as location and temperature, so other areas remain to be researched. The location of a vehicle in our system is anticipated using only a GPS; it would be interesting to add different methods to estimate location, such as Wifi and GSM. Many other ideas could be explored and tested with our system, such the following:

1. Adding a variety of sensors to sense other aspects of the surrounding environment such as humidity, vibration, and pressure.
2. Adding more features that can control the vehicle, such as for starting or turning off the engine, or locking or unlocking the doors.
3. Adding more data storage to the microprocessor to save the recorded data in case of downtime.

Obviously, the use of different types of hardware can play a huge role in the Tracking Unit power consumption. For example, we have used Arduino UNO as a microprocessor but there are many other kinds of Arduino such as the Arduino Mega or Arduino Mini. New approaches to minimizing power can be generated from techniques described in the literature review, such as in [20] or [22].

7 References

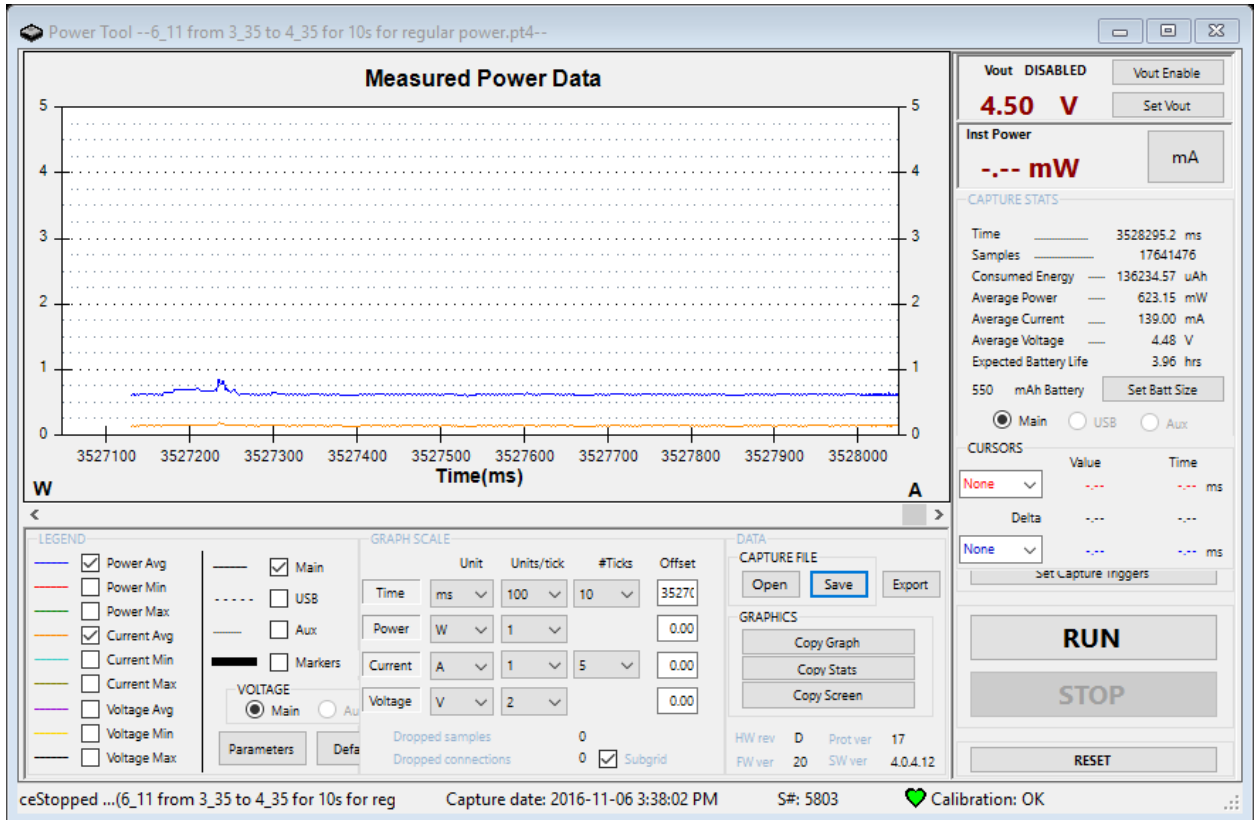
- [1] Piyare, R., & Lee, S. R. (2013). Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing. *arXiv preprint arXiv:1310.2095*.
- [2] Number of vehicles in use worldwide 2014 | Statistic. (n.d.). Retrieved January 11, 2017, from <https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide/>
- [3] Abbasi, A. M. (2016). Categorization and Detection of Energy Bugs and Application Tail Energy Bugs in Smartphones.
- [4] Voelcker, J. (n.d.). 1.2 Billion Vehicles On World's Roads Now, 2 Billion By 2035: Report. Retrieved January 11, 2017, from http://www.greencarreports.com/news/1093560_1-2-billion-vehicles-on-worlds-roads-now-2-billion-by-2035-report
- [5] Kale, S. M., & Kumar, A. Location Tracking and Data Compression for Accurate Energy Efficient Localization using GSM system.
- [6] Murugananham, S., & Mukesh, P. R. (2010). Real time web based vehicle tracking using GPS. *J. World Acad. Sci. Eng. Technol*, 61, 91-99.
- [7] Chadil, N., Russameesawang, A., & Keeratiwintakorn, P. (2008, May). Real-time tracking management system using GPS, GPRS and Google earth. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on* (Vol. 1, pp. 393-396). IEEE.
- [8] Lu, G., & Zeng, W. H. (2014). Cloud computing survey. In *Applied Mechanics and Materials* (Vol. 530, pp. 650-661). Trans Tech Publications.
- [9] Rouse, M. (2015, February 1). What is cloud computing? - Definition from WhatIs.com. Retrieved March 20, 2015, from <http://searchcloudcomputing.techtarget.com/definition/cloud-computing>
- [10] Susanto, Heru, Mohammad Nabil Almunawar, and Chen Chin Kang. "Toward cloud computing evolution: efficiency vs trendy vs security." (2012).
- [11] IaaS, PaaS, SaaS (Explained and Compared). (n.d.). Retrieved February 10, 2017, from <https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/>
- [12] Borja, S., Ruben, M. S., Ignacio, M. T., & Ian, F. (2009). An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 1, 14-22.
- [13] Mell, P. M., & Grance, T. (2011). The NIST definition of cloud computing. *NIST Special Publication 800-145*. doi:10.6028/nist.sp.800-145
- [14] Garude, M., & Haldikar, N. (2014). Real Time Position Tracking System Using Google Maps. *International Journal of Scientific and Research Publications*, 357.

- [15] Bhadane, D. S., Bharati, P. B., Shukla, S. A., Wani, M. D., & Ambekar, K. K. (2015). A Review on GSM and GPS Based Vehicle Tracking System. *International Journal of Engineering Research and General Science*, 3(2), 351-353.
- [16] Khan, A., & Mishra, R. (2012). GPS–GSM based tracking system. *International Journal of Engineering Trends and Technology*, 3(2), 161-164.
- [17] Verma, P., & Bhatia, J. S. (2013). Design and development of GPS-GSM based tracking system with Google map based monitoring. *International Journal of Computer Science, Engineering and Applications*, 3(3), 33.
- [18] Kotte, S., & Yanamadala, H. B. (2013). Advanced Vehicle Tracking System on Google Earth Using GPS and GSM. *international journal of computer trends and technology (IJCTT)– volume, 6*.
- [19] Vigneshwaran, K., Sumithra, S., & Janani, R. (2015). An Intelligent Tracking System Based on GSM and GPS Using Smartphones. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 04(05), 3897-3903.
doi:10.15662/ijareeie.2015.0405016
- [20] Shabna, V. C., Jamshid, K., & Kumar, S. M. (2014, April). Energy Minimization by removing data redundancy in Wireless Sensor Networks. In *Communications and Signal Processing (ICCSP), 2014 International Conference on* (pp. 1658-1663). IEEE.
- [21] Hamad, F., Smalov, L., & James, A. (2009). Energy-aware Security in M-Commerce and the Internet of Things. *IETE Technical review*, 26(5), 357-362.
- [22] Palattella, M. R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L. A., Boggia, G., & Dohler, M. (2013). Standardized protocol stack for the internet of (important) things. *IEEE Communications Surveys & Tutorials*, 15(3), 1389-1406.
- [23] Villas, L. A., Boukerche, A., De Oliveira, H. A., De Araujo, R. B., & Loureiro, A. A. (2014). A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12, 69-85.
- [24] R., & R. (n.d.). D-C Arduino Uno. Retrieved February 2, 2017, from <http://digital.csic.es/bitstream/10261/127788/7/D-c-%20Arduino%20uno.pdf>
- [25] R., & R. (n.d.). D-C Arduino Uno. Retrieved February 2, 2017, from <http://digital.csic.es/bitstream/10261/127788/7/D-c-%20Arduino%20uno.pdf>
- [26] Weranga, K. S. K., Chandima, D. P., & Kumarawadu, S. P. (2012, May). Smart metering for next generation energy efficiency & conservation. In *Innovative Smart Grid Technologies-Asia (ISGT Asia), 2012 IEEE* (pp. 1-8). IEEE.
- [27] L. (n.d.). Geolocation Tracker (GPRS GPS) with SIM908 over Arduino and Raspberry Pi. Retrieved May 25, 2015, from <https://www.cooking-hacks.com/documentation/tutorials/geolocation-tracker-gprs-gps-geoposition-sim908-arduino-raspberry-pi/#atcommands>

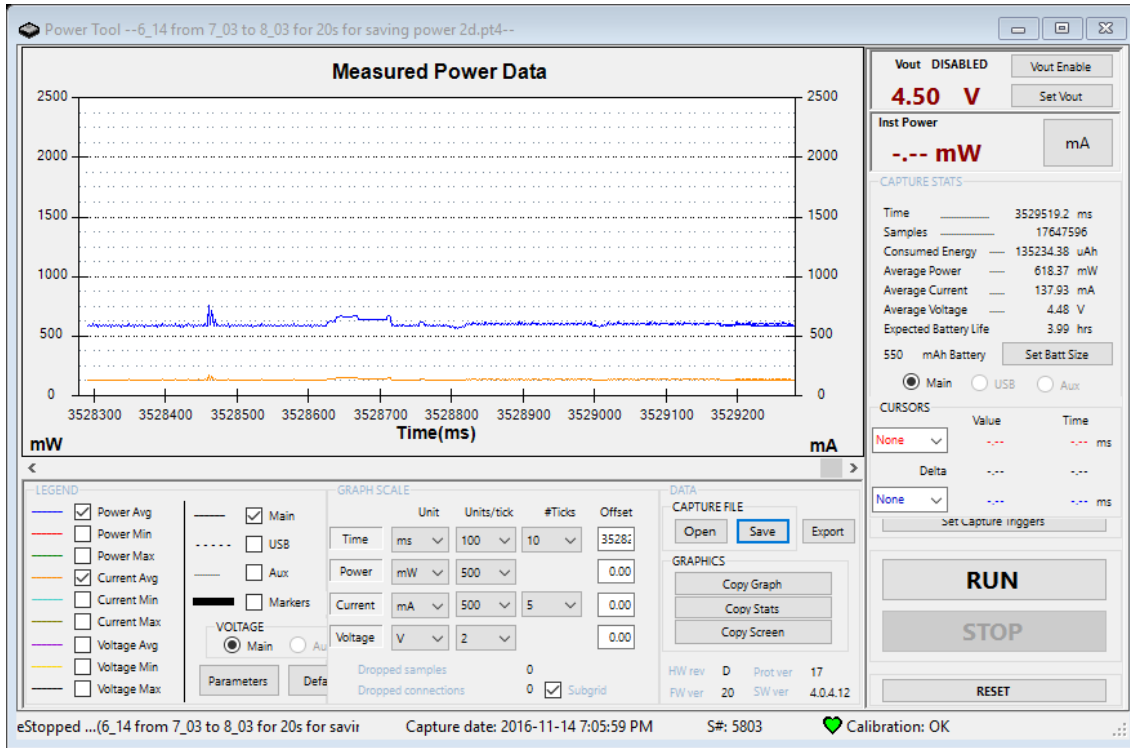
- [28] S. (n.d.). SIM908 Data Sheet. Retrieved October 19, 2016, from <http://image.dfrobot.com/image/data/TEL0051/3.0/SIM908%20datasheet.pdf>
- [29] L. (n.d.). Buy External GPS Antenna SMAM pigtail Online. Retrieved November 02, 2016, from <https://www.cooking-hacks.com/external-gps-antenna-5787>
- [30] L. (n.d.). External GPRS-GSM-UMTS Antenna. Retrieved November 02, 2016, from <https://www.cooking-hacks.com/4g-3g-gprs-gsm-antenna-external>
- [31] Saudi-Arabia. (n.d.). Retrieved January 17, 2017, from <http://www.weatheronline.co.uk/reports/climate/Saudi-Arabia.htm>
- [32] 10 surprising facts about Canadian weather. (2016, June 24). Retrieved January 17, 2017, from <https://www.canadiangeographic.ca/article/10-surprising-facts-about-canadian-weather>
- [33] Narayan, R. (2016, March 28). What is the function of ICSP pins on the Arduino Uno? Retrieved March 27, 2017, from <https://www.quora.com/What-is-the-function-of-ICSP-pins-on-the-Arduino-Uno>

8 Appendix

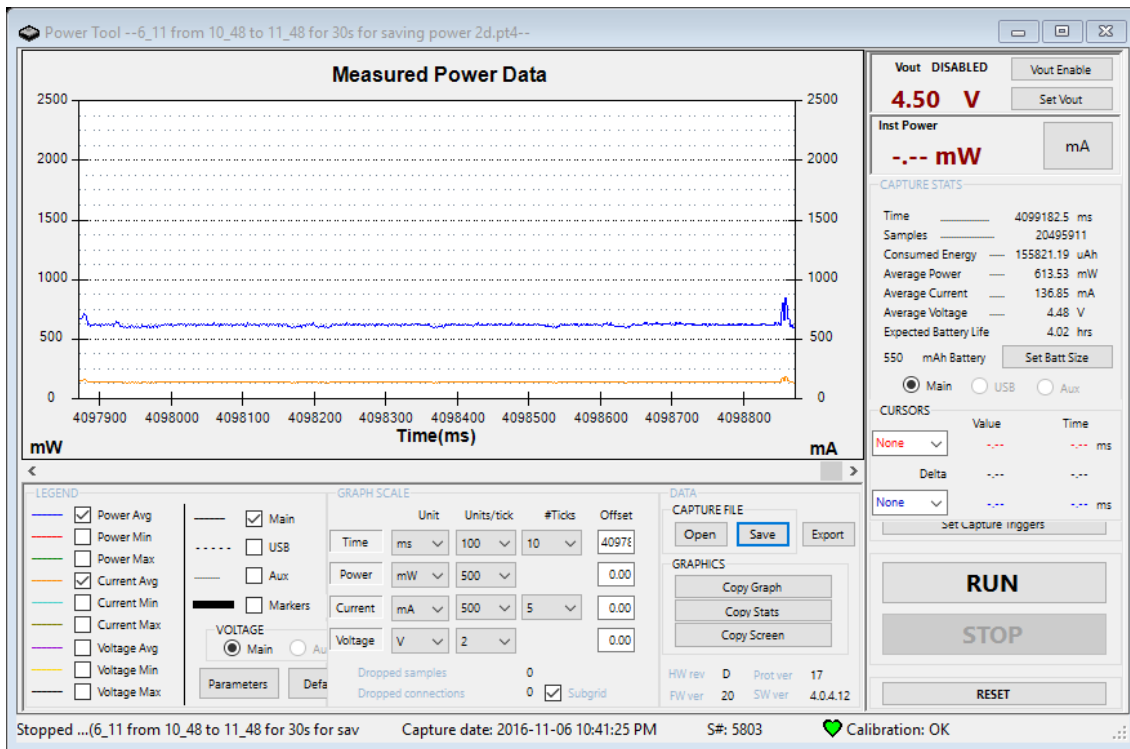
10 second test: for saving power (two digits)



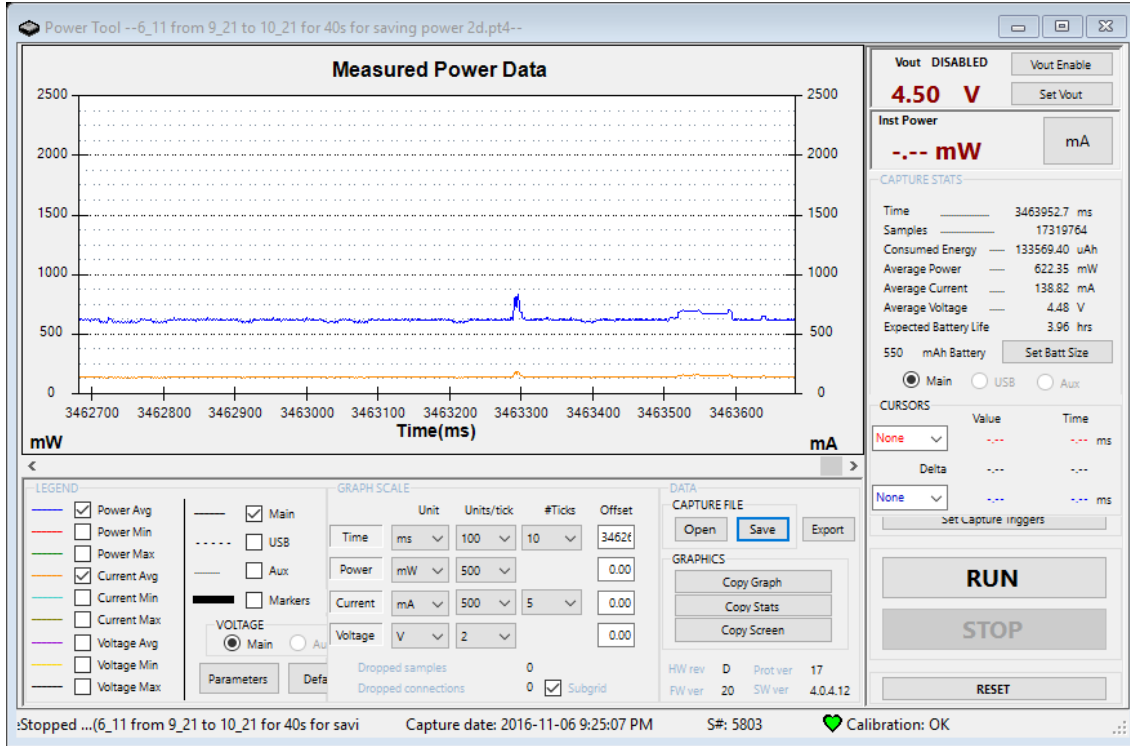
20 second test: for saving power (two digits)



30 second test: for saving power (two digits)



40 seconds test: for saving power (two digits)



50 seconds test: for saving power (two digits)

