

Public Transport Inventory Monitoring System for Developing Countries

Custen Simbarashe Muperi

R141748C



Submitted in partial fulfilment for the degree of

BSc (Hons) TELECOMMUNICATIONS

Department of Applied Physics & Telecommunications in the Faculty of Science and
Technology at the

Midlands State University

Gweru, Zimbabwe

November 2017

Supervisors:

Mr. G Manjengwa and Mr. N Taruvinga

[HTEL 438 Dissertation]

ABSTRACT

This report serves to give an account for a research on a Public Transport Inventory Monitoring System for Developing Countries. The research is based on the current situation in Zimbabwe where the public transport industry is dominated by the informal sector. In this document there is a discussion of the transport industry in Zimbabwe, identification of the problem of accountability of inventory in the current industry, a methodology to help solve the problem and the results of using the method discussed.

This report is a requirement in completion of the Bachelor of Science Honours Degree in Telecommunications at Midlands State University.

DECLARATION

I, **Muperi Custen Simbarashe** hereby declare that I am the sole author of this dissertation entitled “**Public Transport Inventory Monitoring for Developing Countries**”. I authorize Midlands State University to this dissertation only for purposes of scholarly research.

Signature..... Date.....

APPROVAL

This dissertation/thesis entitled “**Public Transport Inventory Monitoring System for Developing Countries**” by **Muperi Custen Simbarashe** meets the regulations governing the award of the degree of **BSc Telecommunications Honours** of the Midlands State University, and is approved for its contribution to knowledge and literal presentation.

Supervisor.....Date.....

ACKNOWLEDGEMENT

Firstly I'd like to thank God for giving me the strength and knowledge to have reached this far in my life. I thank Him for guiding me through this research. I gratefully appreciate Mr. G Manjengwa and Mr. Taruvinga for mentoring me through the research and development of the prototype. I also want to thank the Telecommunications Class of 2017. I want to thank my parents and my siblings for consistently supporting me financially and emotionally. Lastly I'd like to Midlands State University for providing me with the resources and educating me up to this time so achieve completion in this research.

I acknowledge with thanks.

TABLE OF CONTENTS

ABSTRACT.....	2
DECLARATION.....	3
APPROVAL	4
ACKNOWLEDGEMENT.....	5
TABLE OF CONTENTS	6

CHAPTER 1

1.1 INTRODUCTION.....	9
1.2 PROBLEM STATEMENT	9
1.3 MOTIVATION AND BACKGROUND	10
1.4 AIM AND OBJECTIVES OF THE RESEARCH	10
1.5 DISSERTATION LAYOUT	11
REFERENCES.....	12

CHAPTER 2

2.1 INTRODUCTION.....	13
2.2 THE ARDUINO UNO REV3 BOARD	13
2.2.1 Arduino Features	15
2.2.2 Microcontroller – ATmega328P	15
2.2.3 Power.....	17
2.2.4 Input and Output pins	17
2.2.5 Memory	17
2.2.6 Communication	17
2.2.7 Advantages and Disadvantages of Arduino.....	18
2.3 UART GPS NEO 7M-C.....	18
2.3.1 UART GPS Features.....	19
2.3.2 Advantages and Disadvantages of GPS	20
2.4 SIM900 GSM/GPRS SHIELD	20
2.4.1 SIM900 Features	23
2.4.2 Advantages and Disadvantages of GSM	23
2.5 ESP8266 Wi-Fi MODULE	24
2.5.1 ESP8266 Features	25

2.5.2 Advantages and Disadvantages of a Wi-Fi Module.....	26
REFERENCES.....	27

CHAPTER 3

3.1 INTRODUCTION.....	30
3.2 STRUCTURE OF THE SYSTEM	30
3.2.1 System Hardware.....	30
3.2.2 System Software.....	30
3.3 ARDUINO TO GSM SHIELD CONNECTION.....	31
3.3.1 Steps to Connecting SIM900 to Arduino	32
3.4 CONNECTING THE GPS MODULE TO ARDUINO.....	32
3.5 ESP8266 TO ARDUINO CONNECTION.....	33
3.5.1 Initializing the ESP8266	34
3.6 TRANSISTORS	35
3.6.1 Working Principle.....	36
3.7 SEAT SENSORS.....	37
3.8 SOFTWARE DEVELOPMENT	38
3.8.1 Webpage Development	38
3.9 SYSTEM ARCHITECTURE	39
3.9.1 System Connection.....	40
3.10 SYSTEM FLOW CHART	42
REFERENCES.....	43
4.1 INTRODUCTION.....	44

CHAPTER 4

4.2 THE PUBLIC TRANSPORT INVENTORY MONITORING SYSTEM.....	44
4.3 WEBSERVER USER ACCESS PAGE	45
4.4 ROUTE MAPPING	46
4.4.1 Landmarks.....	46
4.4.2 Route Optimization.....	47
4.4.3 On-Route Coordinates.....	48
4.4.3 Off-Route Coordinates	49
4.5 SMS ALERTS	50
4.6 ANALYSIS	50

CHAPTER 2

5.1 INTRODUCTION..... 51

5.2 SYSTEM DRAWBACKS..... 51

5.3 CHALLENGES FACED 51

5.4 RECOMMENDATIONS..... 52

5.5 CONCLUSION 52

APPENDIX..... 53

CHAPTER 1

1.1 INTRODUCTION

Inventory monitoring is a key step in the development of a country's transport industry. Being able to monitor and track a fleet is a practical need for all major players in the transport industry. This concept is also known as Fleet Management. It is defined as a function to try and remove or reduce the risks involved in vehicle investment, improving efficiency, profitability and being compliant with the local legislations [1]. In Zimbabwe, according to the Herald [2], the public transport is chaotic as there is no order and organization. The transport system in Zimbabwe is mainly run by the private sector given permission by the local government to operate on stipulated routes. Organization is one of the key benefits of fleet management among others like: improved time management, reduced maintenance costs, route optimization and customer satisfaction [3]. Route optimization is a huge problem in Zimbabwe, as many commuter omnibus owners are unable to enforce this to their drivers. It is a typical situation in Zimbabwe that commuter omnibus owners dispatch vehicles to drivers and are unaware of the vehicles' activity during the entire day, only to receive the profit at the end of the day. This profit, however is subject to the discretion of the driver and conductor as it is untraceable to know how much was made in the day, how far the vehicle was operated and to which direction was it driven to. Therefore, it is of great importance that the owner of the vehicles may have such insight in the day to day running of the business.

1.2 PROBLEM STATEMENT

Fleet management systems exist in Zimbabwe but are mainly featured in the freight industry. The systems have never been implemented in the common public transport commuter omnibuses also known as "kombis". This is mainly because the fleet management model provided by the suppliers is not specific to the kombi business as well the cost to implement, the suppliers use a 'one size fits all' approach [4]. As of February 2016, there were over 60 000 commuter omnibuses in Zimbabwe, according to the Financial Gazette [5]. Owners of these commuter omnibuses face a lot of challenges in managing their fleet as it is informal. Some of the problems faced by business owners in this industry include [6]: failing to optimize routes, failing to

account of fleet activity during a day, risk to theft and failing to recover, vehicles sustaining wear and tear as damaging routes may be taken and many other problems. At the end of the day the business owner is losing money. The owner of the vehicle has no clear view on what the fleet going through as each day passes by. With all this in mind it is vital that the owner may be able to know how the fleet is behaving during the day that is knowing which routes they travelling, how many passengers are on board and is the vehicle being compliment to the route regulations. This insight will also help to manage finances and reduce the risk of the drivers and conductors from stealing from the business as it is an inevitable truth [7].

1.3 MOTIVATION AND BACKGROUND

Due to the realization of the problems stated above, the researcher has come up with an idea to try solve them. The researcher has taken the situation of an informal public transport service provider who has a fleet of commuter omnibuses that travel in predefined routes every day. The researcher's idea is to help manage this fleet by providing positional tracking and state of the vehicles. The business owner will be able to track their vehicles at any time of the day and get instant notifications when the any misdemeanors have happened. The researcher's idea will help the owner to know how many passengers aboard an omnibus during each trip. The idea is mainly directed to this informal source of public transport. This will however try and curb the problems faced in fleet management.

1.4 AIM AND OBJECTIVES OF THE RESEARCH

The aim of the research is to develop a system that will track the movement of a member of a fleet, know how many passengers are aboard the trip and also track the time taken during the trip. The system is supposed to present the positional data, passenger data and alert the business owner whenever the vehicle goes off-route. The system should allow the business owner to switch off the vehicle remotely using the concept of Internet of Things (IoT).

Objectives of the project:

- To do real-time tracking of the movement of a commuter bus using Global Positioning System (GPS), read the number of passengers onboard and check if vehicle is on-route.
- To plot a real-time map with the positional markers using coordinates received from a GPS module.
- To program an Arduino to send data to a remote webserver and process instructions received from the webserver.

1.5 DISSERTATION LAYOUT

The dissertation was organized as follows:

Chapter 2 – Literature Review. This chapter gives a comprehensive discussion of the components to be used and a discussion of the concept of Fleet Management as presently in other developed countries as well as third-world countries. It will also include the discussion of the applicability of the concept in developing countries. The researcher will exhaust the necessary information behind the concept of Inventory Management for Public Transport providers.

Chapter 3 – Methodology. In this chapter the technical side of implementing the project will be discussed. The components to be used will be discussed in detail and their use or purpose in meeting the objectives of the project. The way in which the components are to be combined and code programming the Arduino will be clearly stated.

Chapter 4 – Results and Analysis. In this chapter the results attained after testing the prototype will be stated. The results will be analyzed and used to determine further recommendations. Some of the results will include the result map, latencies in getting the GPS coordinates as well latencies in the exploration of IoT.

Chapter 5 – Conclusion and Recommendations. This chapter will cover work plan of the research project, the costs of practical implementation, recommendations, conclusion of the project's results and analysis. Areas of further research will be discussed as well.

REFERENCES

1. Spireon. “What is Fleet Management?” Internet: <http://www.spireon.com/fleet-management-blog/what-is-fleet-management/>, [Oct. 2, 2017].
2. The Herald. “Efficient public transport system overdue in Harare.” Internet: <http://www.herald.co.zw/efficient-public-transport-system-overdue-in-harare/>, [Sept. 22, 2017].
3. Market Inspector. “Advantages of Efficient Fleet Management.” Internet: <https://www.market-inspector.co.uk/blog/2015/05/advantages-of-efficient-fleet-management/>, [Sept. 20, 2017].
4. Global Fleet Insights. “Fleet Optimization in Developing Countries - best practices that deliver.” Internet: <https://www.globalfleetinsights.com/fleet-optimisation-developing-countries-best-practices-deliver/>, [Sept. 20, 2017].
5. Financial Gazette. “Kombi ban to affect half a million.” Internet: <http://www.financialgazette.co.zw/kombi-ban-to-affect-half-a-million/>, [Sept. 19, 2017].
6. Ctrack UK. “Fleet Management Problems.” Internet: <https://www.ctrack.co.uk/news-information/telematics-blog/entry/vehicle-tracking/solve-fleet-management-problems.html>, [Sept. 19, 2017].
7. StartupBiz Zimbabwe. “Starting a Minibus/Kombi Transport Business in Zimbabwe and the Business Plan.” Internet: <http://startupbiz.co.zw/starting-minibus-kombi-transport-business-zimbabwe-business-plan/>, [Oct. 1, 2017].
8. *Introducing GPS to Staff*, 1st ed. Fleetmatics Development Limited, 2016, pp. 1-5.
9. *Field Vehicle Fleet Management in Humanitarian Operations: A Case-based Approach*, 2nd ed. Fontainebleau: INSEAD, 2010.

CHAPTER 2

2.1 INTRODUCTION

This chapter will discuss the literature review and theoretical functioning of the electronic components to be used in the Public Transport Inventory System for Developing Countries. The theoretical capabilities of each component and its characteristics will be discussed. Similar systems that have been implemented will be discussed as well and how they function to achieve the fleet management objectives.

2.2 THE ARDUINO UNO REV3 BOARD

The Arduino Uno is an open source microcontroller board based on the ATmega328P microcontroller by Atmel Corporation [1]. Arduino has revolutionized electronics as the Uno is an inexpensive control board, easy to program and can hook up to a wide range of hardware [2]. The Arduino Uno has a combination of digital and analog pins that can be used as inputs or outputs. The Uno has 14 digital pins (pin 0 – 13) of which 6 can be used as Pulse Width Modulation (PWM) outputs, 6 analog input pins (pin 0 – 5), a 16MHz crystal oscillator (ceramic resonator), an ICSP header, a USB connection, a power jack, and a reset button [3][4]. The Arduino is built around the ATmega328P microcontroller thus it has everything to support the microcontroller [5]. It can be powered by a USB cable, a AC-to-DC power adapter or a battery.

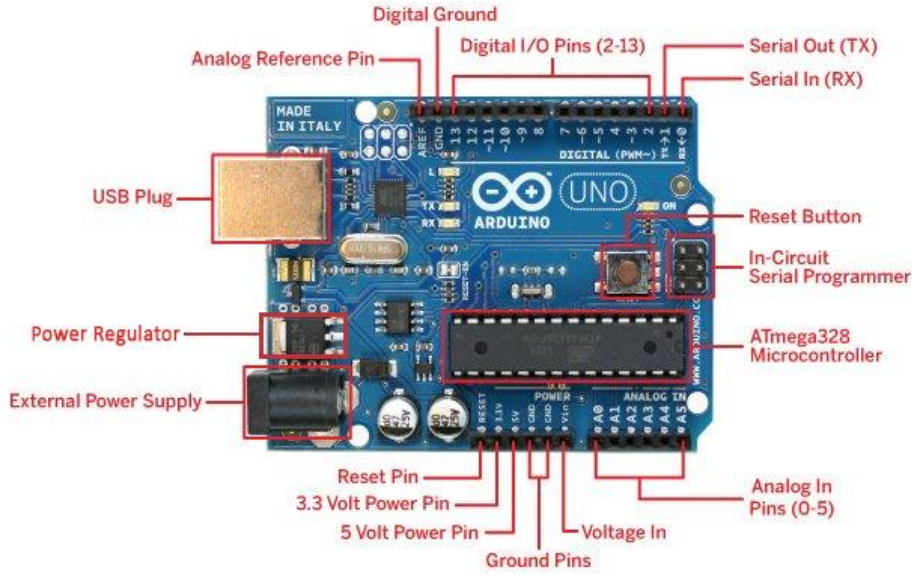


Fig 1 Arduino Uno Rev3

[6]

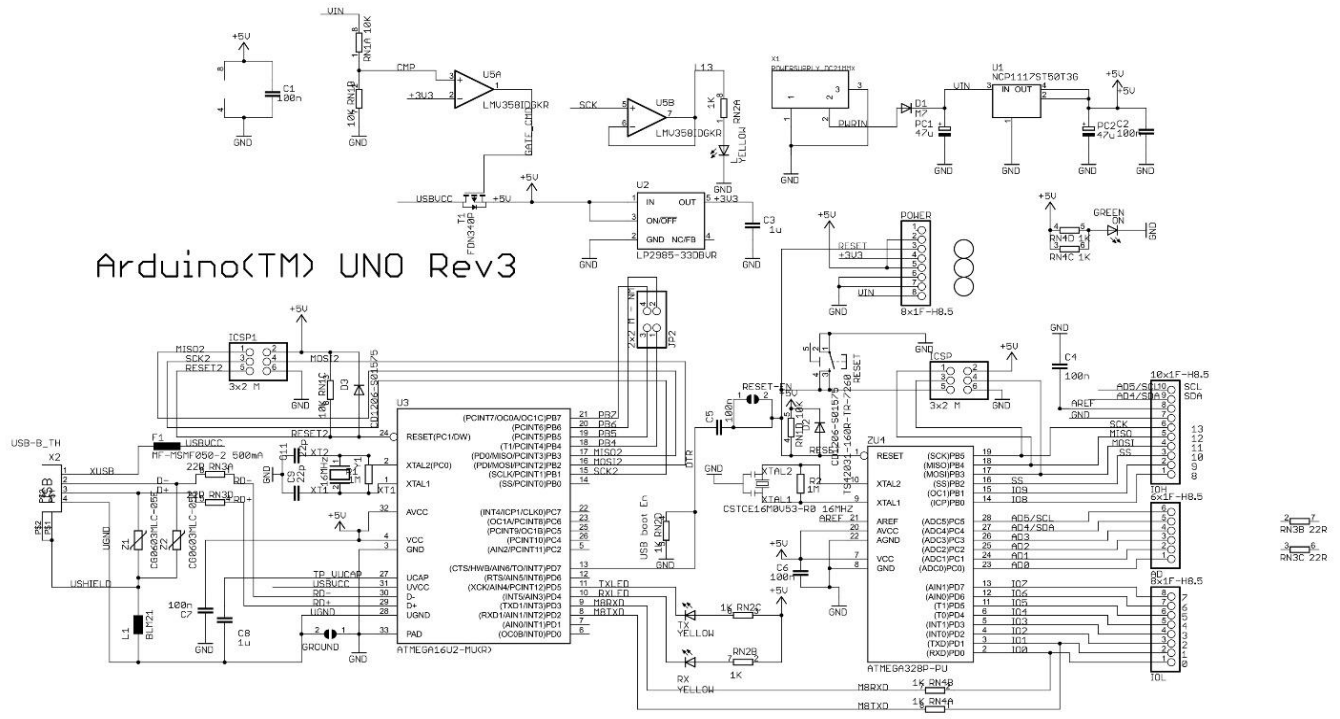


Fig 2 Arduino Uno Rev3 Schematic [7]

2.2.1 Arduino Features

- Microcontroller ATmega328P
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limit) 6-20V
- Digital I/O Pins 14 (of which 6 provide PWM output)
- PWM Digital I/O Pins 6
- Analog Input Pins 6
- DC Current per I/O Pin 20 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32KB(ATmega328P), 0.5KB used by bootloader
- SRAM 2 KB (ATmega328P)
- EEPROM 1 KB (ATmega328P)
- Clock Speed 16 MHz

[8]

2.2.2 Microcontroller – ATmega328P

The ATmega328/P is a low-power CMOS 8-bit microcontroller based on enhanced RISC architecture [9]. It has powerful processing achieving throughputs close to 1 MIPS per MHz by executing powerful instructions in a single clock cycle. The ATmega328/P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 1 serial programmable USARTs , 1 byte-oriented 2-wire Serial Interface (I2C), a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages) , a programmable Watchdog Timer with internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning.



Fig 3 ATmega328/P [10]

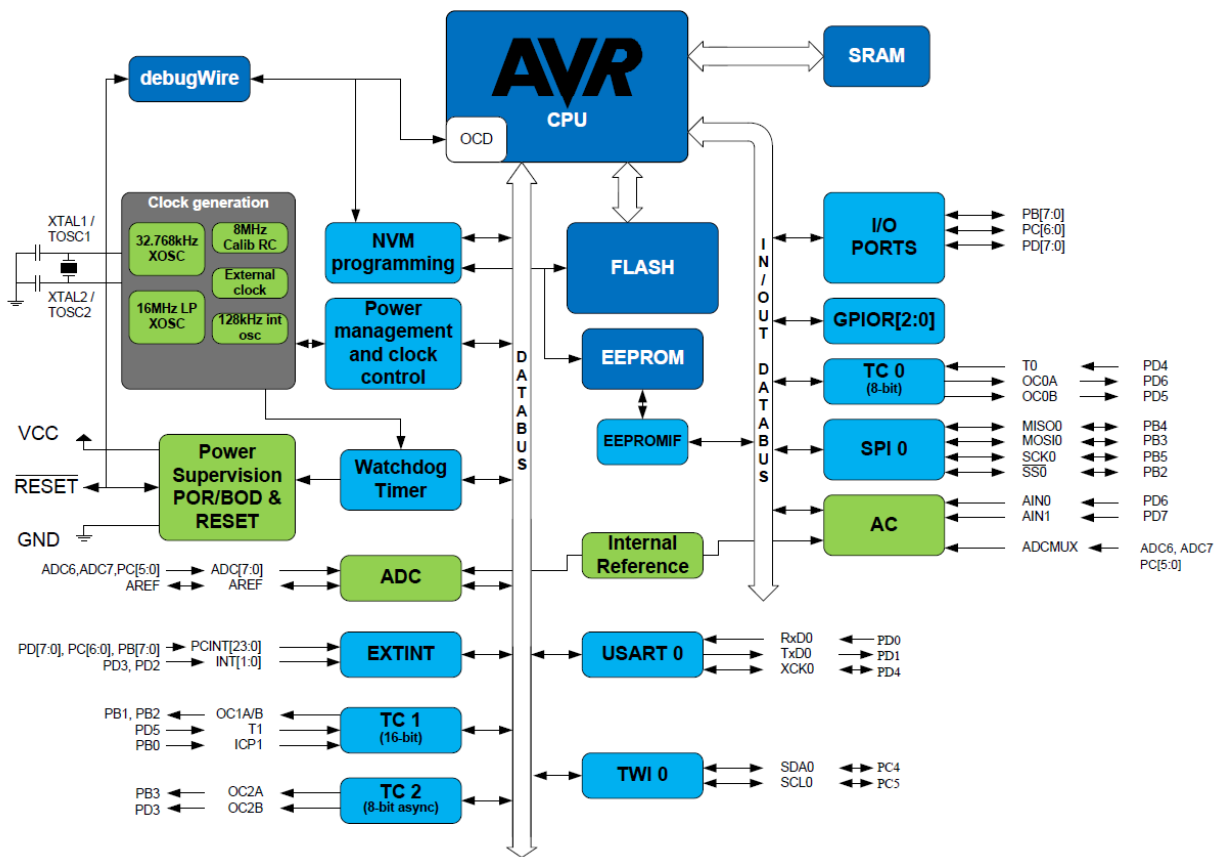


Fig 4 ATmega328P Schematic [9]

2.2.3 Power

The Arduino can be powered by a USB cable, a AC-to-DC power adapter or a battery. The power supply is chosen automatically. The Arduino's operating voltage is from 6V to 20V, though the recommended is 7V to 12V, voltage below 7V will result with an output voltage of less than 5V and voltage above 12V may cause the power regulator to overheat. The **Vin** pin on the board is used when powering from a battery where it is connected to the **Vin** and **Gnd** pins. The Arduino has 2 voltage outputs: 5V and 3.3V.

2.2.4 Input and Output pins

Each of the 14 digital pins can be used as inputs or outputs operating at 5V, each can receive or supply up to 40mA of current and has an internal pull-up resistor of 20 - 50kOhms. Pins **0** and **1** are used as transmit and receive pins for serial TTL data. **2** and **3** are configured to trigger an interrupt on a low when called to, also known as external interrupts. Pins **3**, **5**, **6**, **9**, **10** and **11** provide PWM outputs when programmed with the **analogWrite()** function. Pins **10 (SS)**, **11 (MOSI)**, **12 (MISO)**, **13 (SCK)** support Serial Peripheral Interface (SPI) communication [11]. The Arduino has 6 analog output pins labelled **A0** through to **A5** which measure from ground to 5V. Pins **A4(SDA)** and **A5(SCL)** support TWI communication.

2.2.5 Memory

The Arduino uses the built-in memory of the ATmega328P which is stated above. It has 32KB of memory, 2KB of SRAM and 1KB of EEPROM.

2.2.6 Communication

The Arduino has features to communicate with a computer, another Arduino and other microcontrollers. The ATmega328P supports TTL serial communication. Arduino has SPI and TWI available through its pins as well a **SoftwareSerial** library which allows for serial communication with any of the Arduino Uno's digital pins. A USB connection from a computer is provided.

2.2.7 Advantages and Disadvantages of Arduino

Advantages	Disadvantages
Low cost	Too easy to use (user does not learn much)
Easy to program	Flooded code (user will not learn to program)
Large community (ease of access to help and assistance through forums)	Not efficient in power sensitive or time sensitive projects.
Fast prototyping (easy assembling of components around the Arduino)	Limitations to customized use
Examples of code (comes with libraries with examples)	Uncompromisable structure

Table 1 Advantages and Disadvantages of Arduino [12][13]

2.3 UART GPS NEO 7M-C

The UART GPS NEO is a Global Positioning System (GPS) module which uses a high gain active antenna. It can be used as a GPS navigator and for quadcopter positioning [14].

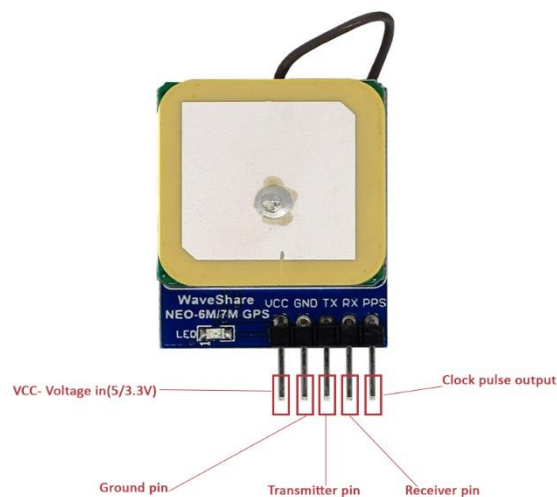


Fig 4 NEO 7M-C [15]

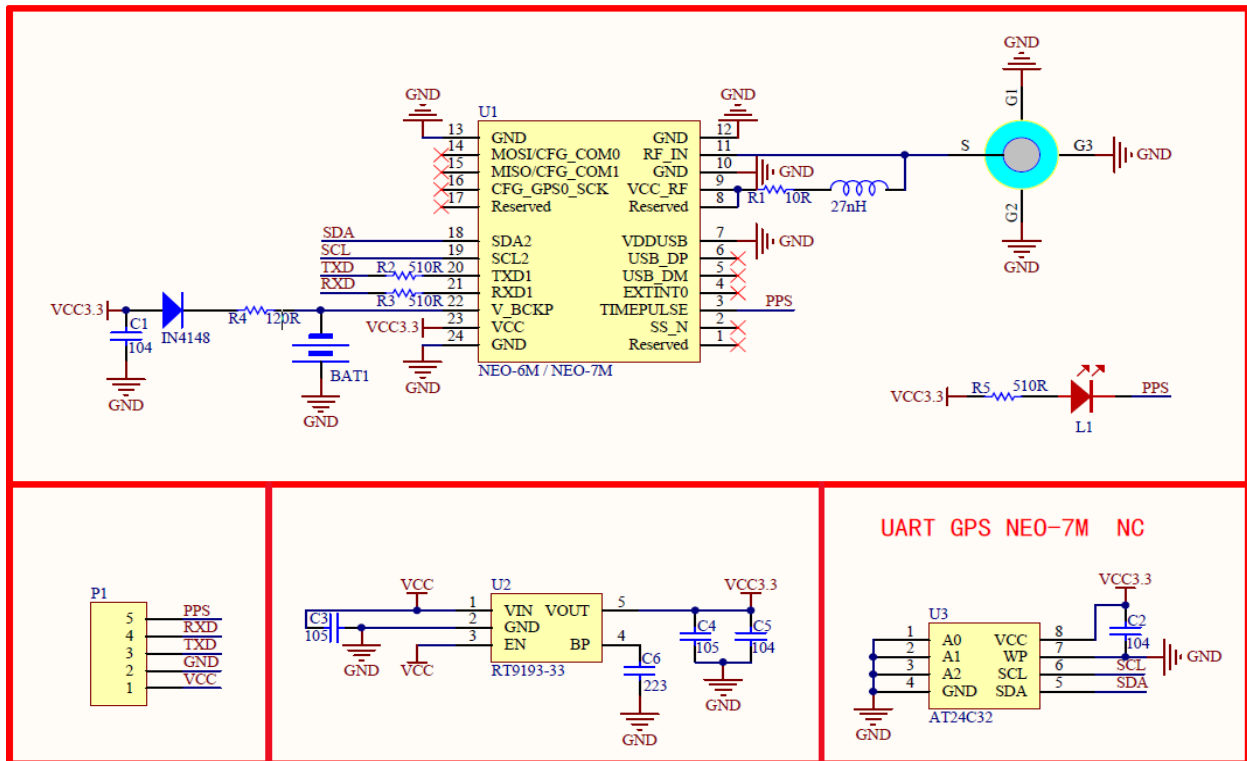


Fig. 5 NEO 7M Schematic [tbw]

2.3.1 UART GPS Features

- Receiver type: 56 channels, GPS L1(1575.42Mhz) C/A code, SBAS:WAAS/EGNOS/MSAS
- Horizontal position accuracy: 2.5mCEP (SBAS:2.0mCEP)
- Navigation update rate: 10Hz maximum (1HZ default)
- Capture time: Cool start: 27s (fastest) ; Hot start: 1s
- Tracking & Navigation sensitivity: -162dBm
- Communication protocol: NMEA(default) / UBX Binary
- Serial baud rate: 4800, 9600(default), 19200, 38400, 57600, 115200, 230400
- Operating temperature: -40°C ~ 85°C
- Operating voltage: 2.7V~5.0V(power supply input via VCC)
- Operating current: 35mA
- TXD/RXD impedance: 510 Ohms

2.3.2 Advantages and Disadvantages of GPS

Advantages	Disadvantages
Accessible anywhere on the globe	Full functionality not always guaranteed
Free of charge	Very power consuming
Calibration and improvements by the US Government	Does not penetrate through certain obstacles and buildings.
Good navigation tool in water bodies	No control over malfunctions
Useful tool for tourists	

Table 2 Advantages and Disadvantages of GPS [17][18]

2.4 SIM900 GSM/GPRS SHIELD

The SIM900 is a Quad-band GSM/GPRS module which embraces the Surface-mounted Technology (SMT) [19]. SMT is the method of mounting components right on the top of Printed Circuit Boards (PCB), which became popular in the 1980's [20]. It is, however capable of sending and receiving SMS, calling and receiving calls as well connecting to the internet through GPRS. The SIM900 works on the following frequencies: DCS 1800MHz, EGSM 900MHz, GSM 850MHz and PCS 1900MHz [21]. It is fused with TCP/IP protocol with extended TCP/IP AT commands being customized for users to easily communicate with the device [21]. The SIM900 is designed to consume very little power but achieve high throughputs, in sleep mode it uses current as little as 1.5mA. It can be powered by a 12V AC-DC adapter or through a 5V VCC which can be from the Arduino though the GND has to be shared so as to have a common ground.

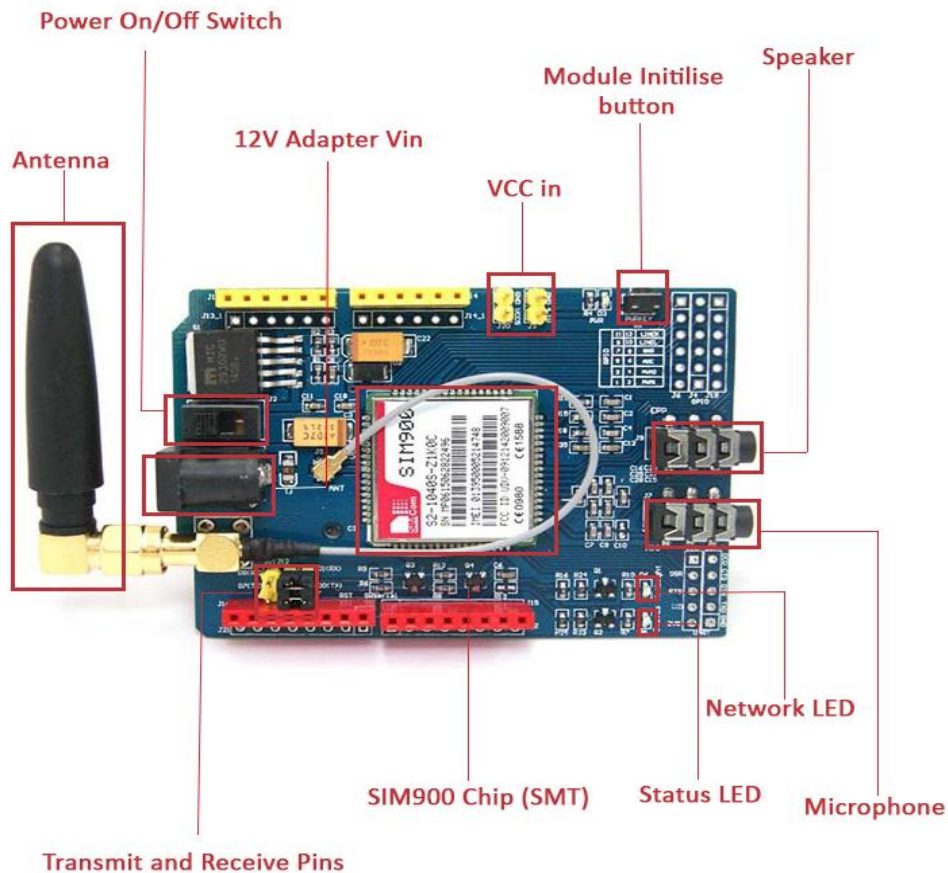


Fig 5 SIM900 [22]

The SIM card for this module will be placed under the shield. After turning on the module from the power on/off switch as shown in Fig 5 above, the functionalities will have to be initialized by pressing and holding the initialize button for 2 seconds. The **Status LED** and the **Network LED** will switch on as soon as the initialize button is pressed, the **Status LED** will remain on and the **Network LED** will then blink off and on signifying that the module has connected to the SIM card's network. From here the GSM module will be active for use. A microphone and speaker can be connected to the module at the respective ports. The functional diagram showing the main components of module is shown in Fig. 6 below.

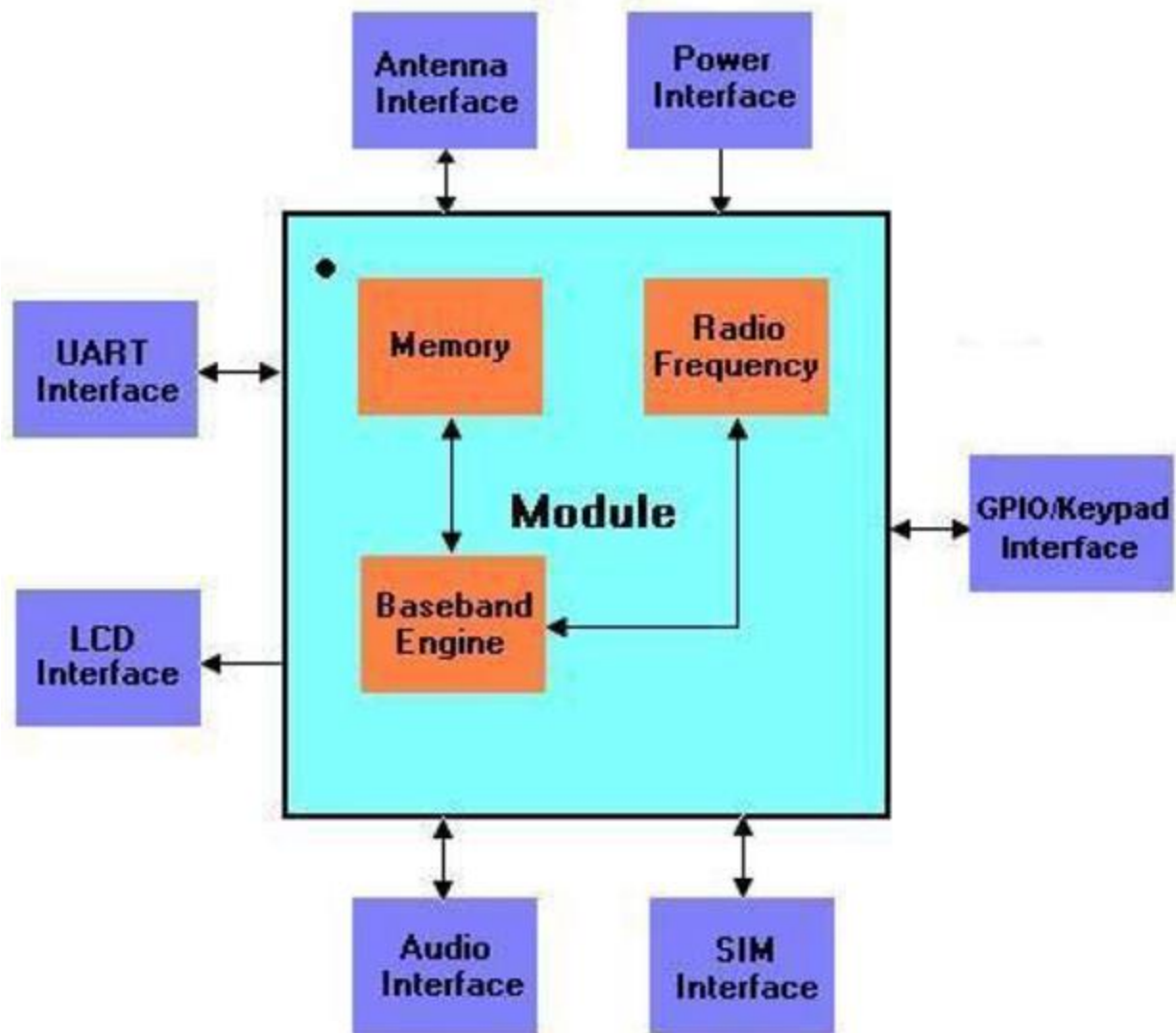


Fig. 6 SIM900 Functional Diagram [21]

2.4.1 SIM900 Features

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
 - Class 4 (2 W @850/ 900 MHz)
 - Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24 * 24 * 3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range: 3.4 to 4.5 V
- Low power consumption
- Operation temperature: -30 °C to +80 °C

[19]

2.4.2 Advantages and Disadvantages of GSM

Advantages	Disadvantages
Extensive coverage globally	Expensive to service providers to run
High transmission quality	Expensive to use to client
Cheap to acquire	Bandwidth lag as it used Frequency Time Division Multiple Access (FTDMA)
Efficient	Electronic Interference with other electronic components

Table 3 Advantages and Disadvantages of GSM. [23][24]

2.5 ESP8266 Wi-Fi MODULE

A self-contained system on a chip greatly integrated with TCP/IP protocol stack to provide the service of connecting a microcontroller to an available Wi-Fi network [25]. An ESP8266 is the most efficient, impressive, power conservative and low cost Wi-Fi module that brings Wi-Fi capabilities to a microcontroller through an UART serial connection [26]. The ESP8266 fully implements the concept of Internet of Things (IoT). IBM defines IoT as “The concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices” [27]. It is a networking system connecting mechanical machines, electronic devices, objects, animals and people all with unique identifiers such that all these communicate with each other even without the interference of a human [28]. It insights that **anything** be given an IP address and it is connected to the entire world, thus there is global remote access.

Internet of Things is the future of technology and the ESP8266 makes prototyping part of the future. It can be applied to any electronic system. With the latest upgrades of Wi-Fi, the ESP8266 stands a pillar in electronics. It features an 802.11 b/g/n protocol and Wi-Fi Direct (P2P).

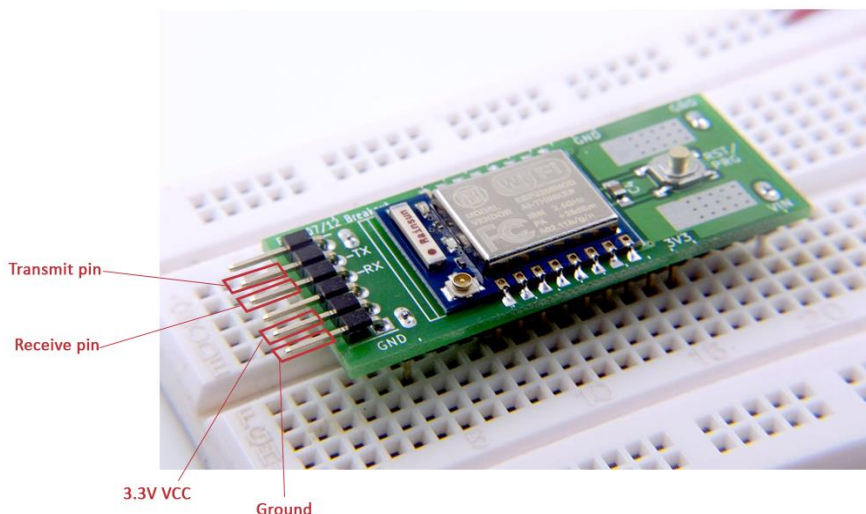


Fig. 7 ESP8266 Wi-Fi Module [29]

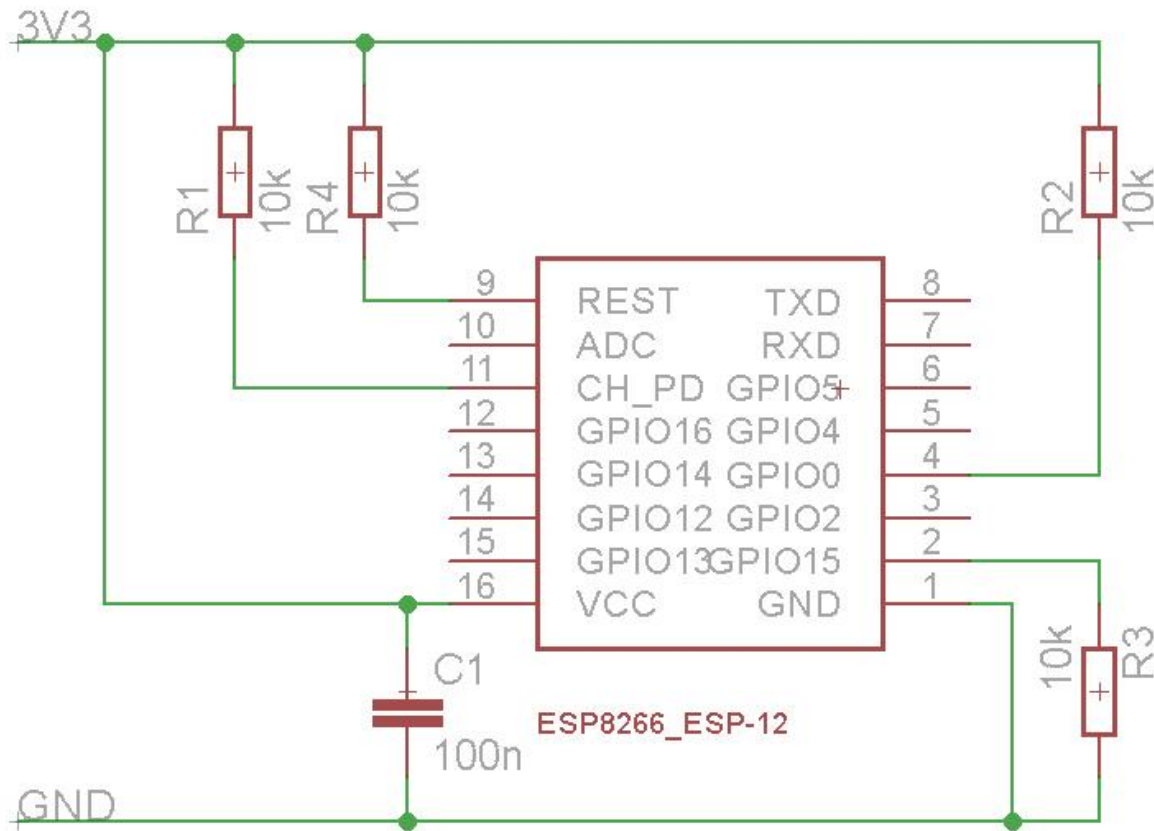


Fig. 8 ESP8266 Schematic [30]

2.5.1 ESP8266 Features

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <math><10\mu\text{A}</math>
- 1MB Flash Memory
- Integrated low power 32-bit CPU could be used as application processor

- SDIO 1.1 / 2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

[25]

2.5.2 Advantages and Disadvantages of a Wi-Fi Module

Advantages	Disadvantages
Easing wireless development	Short range mobility
Mobility	
Remote access	

Table 4 Advantages and Disadvantages of Wi-Fi Module.

REFERENCES

10. Arduino. “Arduino Uno Rev3”. Internet: <https://store.arduino.cc/usa/arduino-uno-rev3/>, [Oct. 3, 2017].
11. MIT Technology Review. “Arduino Uno.” Internet: <https://www.technologyreview.com/s/422845/arduino-uno/>, [Oct. 3, 2017].
12. *Arduino Uno Datasheet*. Farnell, 2016, pp. 1.
13. Trossen Robotics. “Arduino Uno Rev3 Microcontroller.” Internet: <http://www.trossenrobotics.com/p/arduino-uno.aspx>, [Oct. 2, 2017].
14. *Arduino Uno Datasheet*. Farnell, 2016, pp. 2.
15. Robomart. “Arduino Uno R3 Image.” Internet: <https://www.robomart.com/arduino-uno-online-india>, [Sept. 19, 2017].
16. *Arduino Uno Rev3 Schematic*. Arduino CC, 2015, pp. 1.
17. Arduino. “Arduino Uno Rev3 - Technical Specifications”. Internet: <https://store.arduino.cc/usa/arduino-uno-rev3/>, [Oct. 3, 2017].
18. *ATmega328/P Datasheet Complete*, 1st ed. Atmel, 2016, pp. 1, 9
19. Amazon. “ATMEGA328P-PU with Arduino Bootloader – Uno Image.” Internet: <https://www.amazon.com/ATMEGA328P-PU-with-Arduino-Bootloader-Uno/dp/B007SH0D0A>, [Sept. 19, 2017].
20. *Arduino Uno Datasheet*. Farnell, 2016, pp. 2-3.
21. Engineer Experiences. “Pros and Cons of Using Arduino.” Internet: <http://engineerexperiences.com/advantages-and-disadvantages.html>, [Oct. 3, 2017].
22. Electronicsforu. “Advantages and Disadvantages of Using an Arduino.” Internet: <http://forum.electronicsforu.com/forum/technologies-work/the-brain-microcontrollers-microprocessors/development-boards/arduino/33135-advantages-and-disadvantages-of-using-an-arduino>, [Oct. 3, 2017].
23. Waveshare. “UART GPS Module, NEO-7M-C onboard, straight/vertical pinheader.” Internet: <https://www.waveshare.com/uart-gps-neo-7m-c-b.htm>, [Oct. 3, 2017].
24. Global Free Shipping. “Images.” Internet: http://img.dxcdn.com/productimages/sku_312527_1.jpg, [Oct. 3, 2017].
25. *UART GPS NEO-7M-C User Manual*. Waveshare, 2016, pp. 1.

26. RF Wireless World. "Advantages of GPS | Disadvantages of GPS." Internet: <http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-GPS.html>, [Oct. 3, 2017].
27. Rose India. "Advantages and Disadvantages of Global Positioning System." Internet: <https://www.roseindia.net/services/trackingssystem/advantaesanddisadvantagesofgps.shtml>, [Oct. 3, 2017].
28. *The GSM/GPRS Module for M2M applications*. Propox, 2016, pp. 1.
29. Radio Electronics. "What is SMT Surface Mount Technology – Tutorial." Internet: <http://www.radio-electronics.com/info/data/smt/what-is-surface-mount-technology-tutorial.php>, [Oct. 3, 2017].
30. *Hardware Design: SIM900_HD_V1.01*. Shanghai: SIMCom Wireless Solutions Ltd, 2009.
31. Invent Electronics "Images." Internet: <https://www.inventelectronics.com/product/sim-900/>, [Oct. 3, 2017].
32. Techwalla. "Advantages and Disadvantages of GSM." Internet: <https://www.techwalla.com/articles/the-advantages-and-disadvantages-of-gsm>, [Oct. 3, 2017].
33. RF Wireless World. "Advantages of GSM | Disadvantages of GSM." Internet: <http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-GSM.html>, [Oct. 3, 2017].
34. SparkFun. "WiFi Module – ESP8266." Internet: <https://www.sparkfun.com/products/13678#description-tab>, [Oct. 3, 2017].
35. *ESP8266 WiFi Module Quick Start Guide*. Kiwicon & Rancidbacon, 2016, pp. 1.
36. IBM. "What is the Internet of Things – Blog." Internet: <https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/>, [Oct. 3, 2017].
37. TechTarget IoT Agenda. "Internet of Things (IoT)." Internet: <http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>, [Oct. 3, 2017].
38. Tindie. "ESP8266 (ESP-07/12) Full I/O Breadboard Adapter." Internet: <https://www.tindie.com/products/Ba0sh1/esp8266-esp-0712-full-io-breadboard-adapter/>, [Oct. 3, 2017].

39. ESP8266 GitHub. “ESP8266 Arduino Core.” Internet:

<http://esp8266.github.io/Arduino/versions/2.1.0/doc/boards.html>, [Oct. 3, 2017].

CHAPTER 3

3.1 INTRODUCTION

In this chapter there will be a discussion of the hardware connections of the components so as to bring out the actual system. All subsystems are to be discussed and how they have been implemented to achieve the objectives of the project. Flow chart shall be presented clearly describing the stages to which the system goes through as it executes its function.

3.2 STRUCTURE OF THE SYSTEM

3.2.1 System Hardware

The system consists of the following hardware components:

1. Arduino Uno
2. UART GPS NEO 7M-C Module
3. SIM900 GSM Shield
4. ESP8266 mini Wi-Fi Module
5. Buzzer
6. Push Button
7. Transistors
8. Resistors
9. SIM card
10. Laptop

3.2.2 System Software

1. Arduino Serial Monitor
2. XAMPP Webserver
3. Google Chrome browser

3.3 ARDUINO TO GSM SHIELD CONNECTION

A GSM Shield is traditionally just connected by mounting it on top of the Arduino [1]. In this project it was vital to separate the two such that it can be easily disconnected and reconnected. The connection used in this project uses the transmit pin of the GSM (**TXD**) connecting to pin 2 of the Arduino and the receive pin (**RXD**) to pin 3 of the Arduino [2]. The GSM Shield uses an external 12V AC-to-DC power adapter as powering from the Arduino causes hiccups and the module will shut down unexpectedly. Due to the external power supply a ground cable should be connected from the Arduino to the shield so as to initiate same ground potential between the two devices.

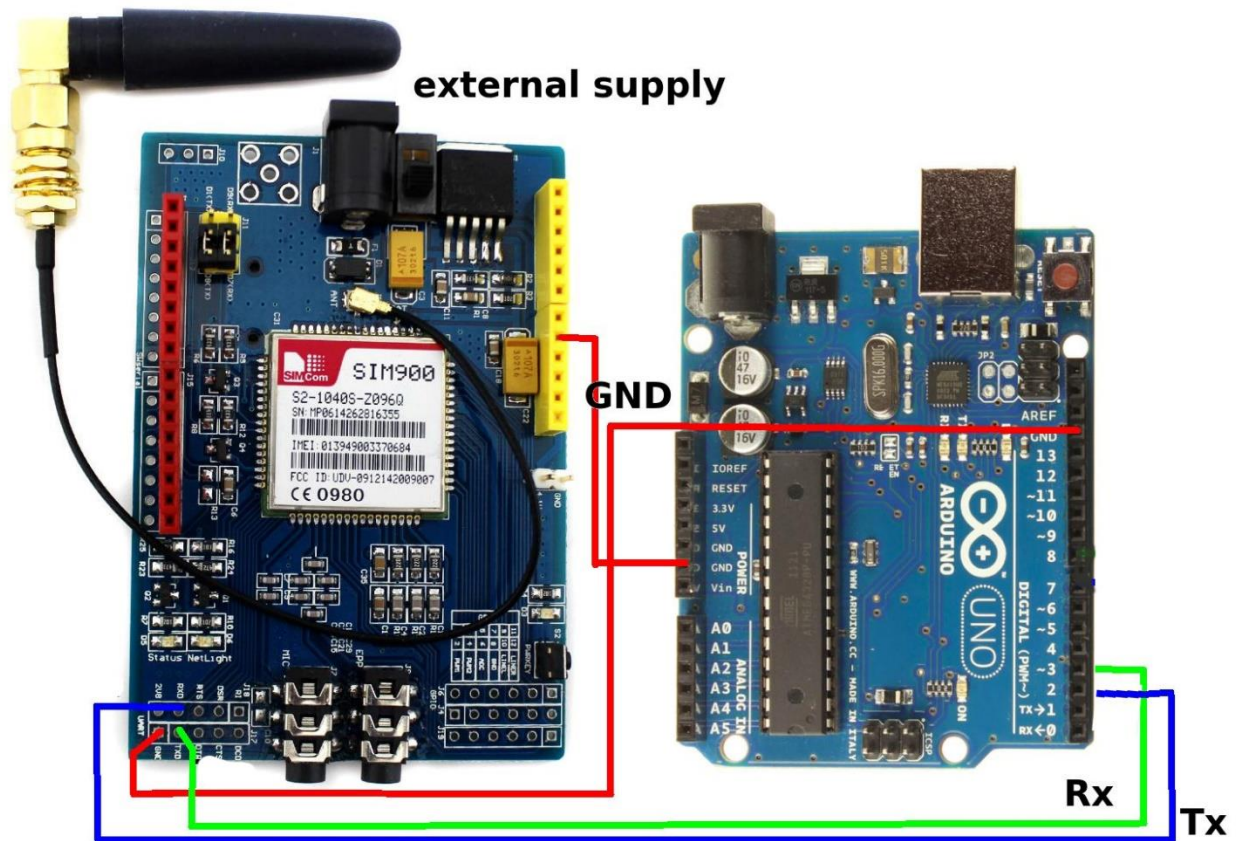


Fig 3.1 Arduino to SIM900 connection [3]

The GSM uses a library in the Arduino code which is then used to talk to the GSM in its language (i.e. AT commands). A library called **GSM SHIELD** is used in the code [4]. The GSM module uses a baudrate of 9600.

3.3.1 Steps to Connecting SIM900 to Arduino

The SIM900 is connected to the Arduino in the following steps:

1. Attach the SIM900 transmit pin, **TXD** to digital pin **2** of the Arduino.
2. Attach the SIM900 receive pin, **RXD** to digital pin **3** of the Arduino.
3. Connect the ground pin of the SIM900 to the ground pin of the Arduino.
4. Insert a standard size SIM card at the bottom of the SIM900 board.
5. Plug in the 12V AC-to-DC power to the board.
6. Toggle the switch next the power slot towards the SIM900 board, this should light up the status light.
7. Press and Hold the initialize button for about 2 seconds, this should trigger the status LED of the SIM900 and also if network is available the network should consistently blink.
8. Send commands to initialize the GSM.

AT commands are then entered so as to confirm if the SIM900 is communicating with the Arduino via the two pins connected. A test command is **AT** which should respond with **OK** if the connection is working.

3.4 CONNECTING THE GPS MODULE TO ARDUINO

The NEO 7M-C is quite easy to connect to the Arduino. It has 5 pins but only 4 are of main importance [5], the 4 essentials pins are **TX, RX, VCC and GND**, the extra pin is **PPS** which is used to measure the clock speed. The terminology in the pins is by default. The module only works properly at a baudrate of 9600 though it has capabilities of other rates like 115200 [6]. In this project the **TX** was connected to digital pin 10 of the Arduino and **RX** to pin 11. The **VCC** used was 5V which is supplied by the Arduino and the ground is from the Arduino as well.

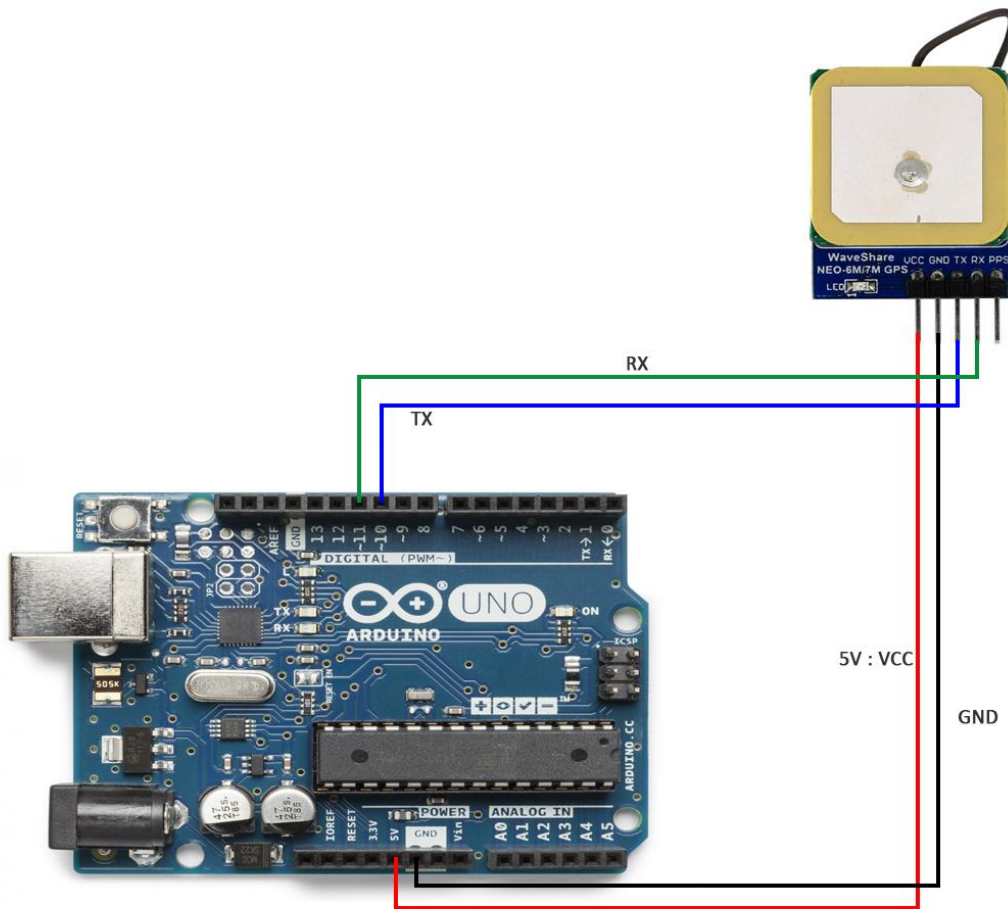


Fig 3.2 GPS to Arduino connection [7][8]

After connecting like this it is essential to use an Arduino library so as to get coordinates from the GPS module. **TinyGPSPlus** library was the one used in this project which makes it easier to consistently collect data from the GPS module [9].

3.5 ESP8266 TO ARDUINO CONNECTION

This connection is a very sensitive one as the ESP8266 is quite specific when it comes to its connection. The ESP8266 has 5 useable pins in this project which are **UTXD**, **URXD**, **VCC**, **GND** and **CH_PD**. The **VCC** supplied to this device should be 3.3V anything higher will fry the board, thus the connection is delicate [10]. The devices are connected as follows:

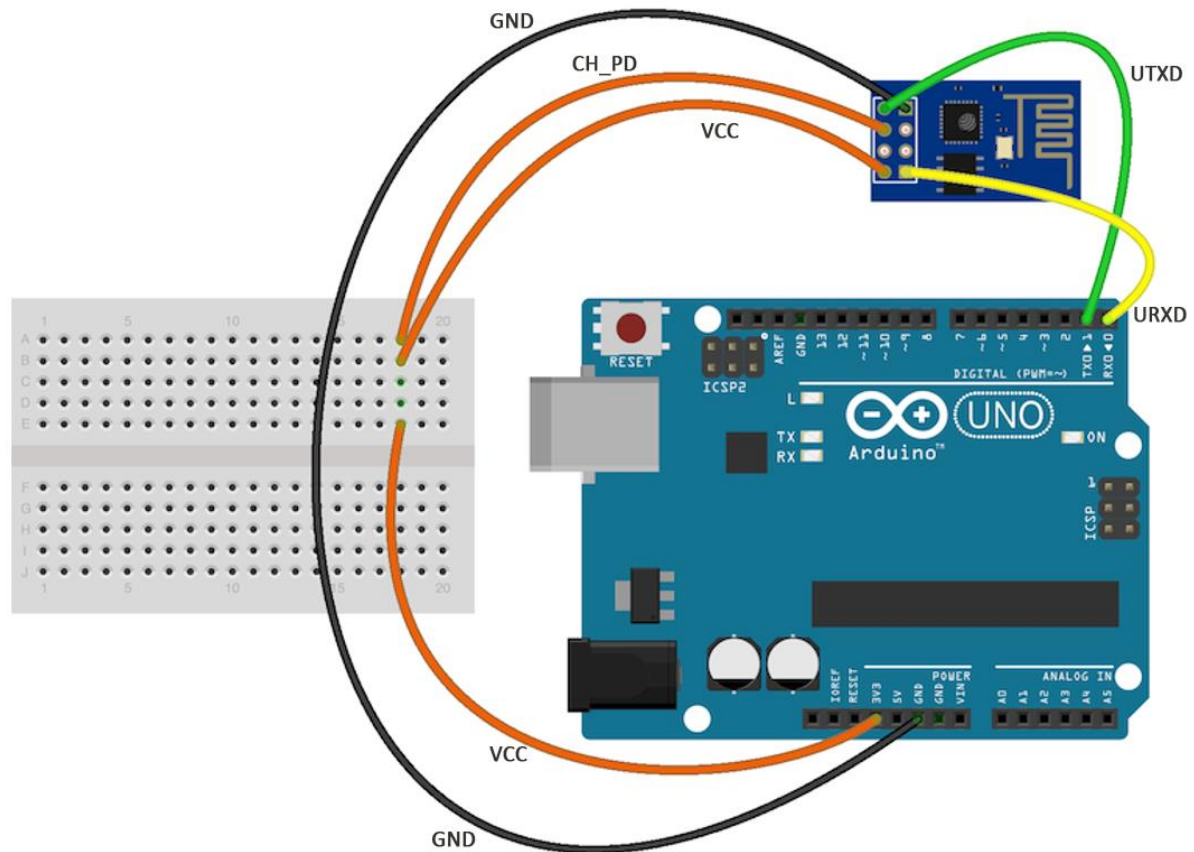


Fig 3.3 ES8266P to Arduino connection [10]

3.5.1 Initializing the ESP8266

The ESP8266 does not need any libraries so as to run but will need **AT** commands to initialize and do the recommended tasks.

Command	Response	Function
AT	OK	Test if AT system works correctly
AT+CWMODE=mode	OK	Set Access Point's info which will be connected by ESP8266

Command	Response	Function
AT+CWQAP	OK	Disconnect from AP that ESP8266 is currently connected to.
AT+RST	OK	Reset the ESP8266.
AT+CWJAP="access point name", "password"	OK	Connect to defined access point.
AT+CIPMUX=1	OK	Allow multiple connections to the ESP(maximum of 4)
AT+CIPSERVER=1, 80	OK	Start ESP as server at port 80

Table 3.1 ESP8266 AT commands [11]

3.6 TRANSISTORS

In this project a Bipolar Junction Transistor are used to work as a switch to toggle the buzzer on and off. A BJT is a simple three, terminal, three layer and two junction semiconductor device. Both types of Bipolar Junction Transistors, Negative-Positive-Negative (NPN) and Positive-Negative-Positive can be used as switches [12] but in this project the NPN was used.



Fig 3.4 Negative-Positive-Negative Transistor [12]

3.6.1 Working Principle

For current to flow from the collector to the emitter the transistor should be *saturated*. This saturation can be achieved if there's a little current through the base. Without this current the transistor is deemed to be in *cutoff* and current will not flow from the collector to the emitter. As used in this project the base of the transistor will be connected to a digital pin on the Arduino thus the pin can control current to either pass or not through from the collector to the emitter. The collector will be connected to the VCC source and the emitter to the positive pin of the buzzer.

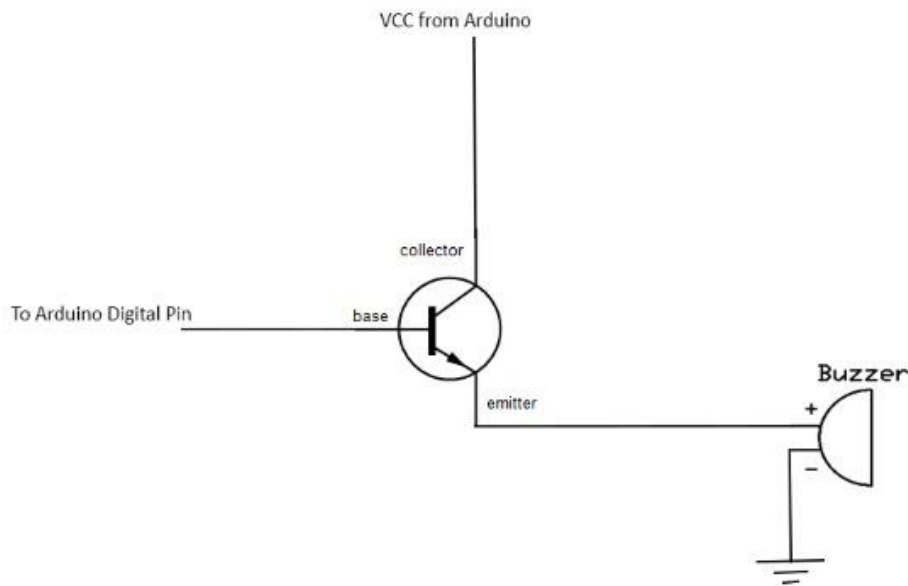


Fig 3.5 NPN as used in project [13]

3.7 SEAT SENSORS

The prototype simulates counting passengers onboard the vehicle. To achieve this the researcher designed custom sensors to explain the concept. The digital pins from the Arduino are used to get values from sensors that detect the state of the seats that is either active or inactive. The design of the sensors uses the simple configuration of a switch used to complete a circuit with the use of two copper wire coils. One coil comes from the 5V VCC input (positive coil) and the other from a resistor then to the ground (negative coil). A digital pin will be connected between the resistor and the negative coil. The two coils are to be in contact when the seat is said to be sat on thus giving a digital **HIGH** value to the digital pin at the Arduino. When a **HIGH** value is detected by the Arduino it increments to the number of active seats which is a variable. There is a total of three sensors in the prototype.

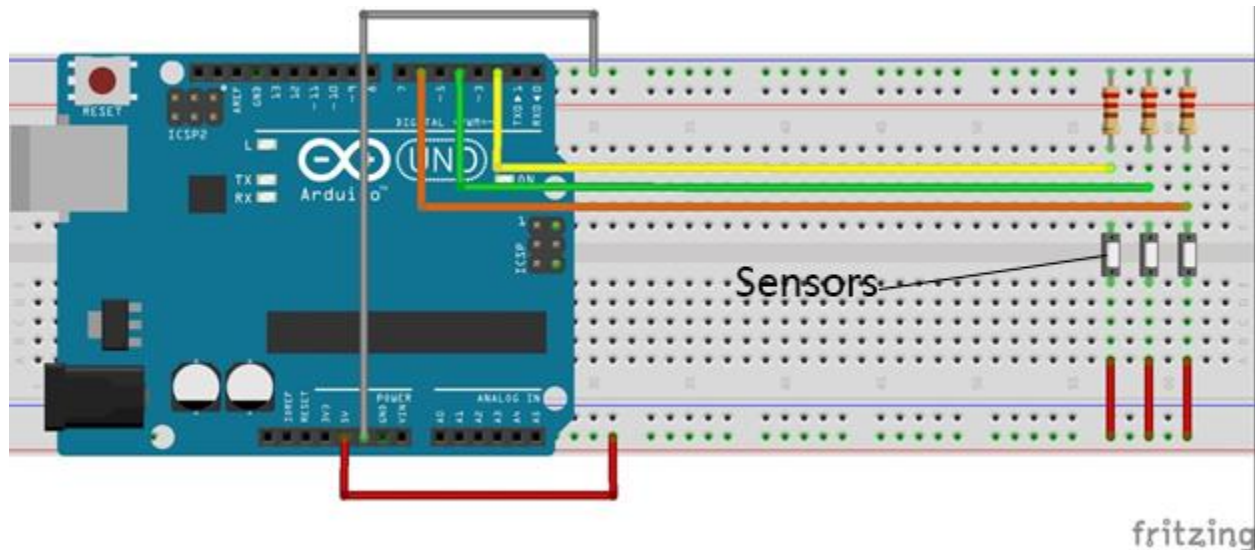


Fig 3.6 Seat Sensors Circuit

The Arduino loop repeatedly runs the function of reading the digital inputs from the sensors, thus having a recent reading after every cycle.

3.8 SOFTWARE DEVELOPMENT

The data collected from the hardware stated above is sent to a software side of the project. The software resides on a laptop. A webserver hosts web pages which are used to send data received from the hardware to a database and also to read data from the database, process it and present it to the user interface. The webserver used is XAMPP v3.2.2. XAMPP is an open source cross-platform webserver with Apache distribution containing MySQL, MariaDB, PHP and Perl [14]. The webpages are in Hypertext Preprocessor (PHP) language and JavaScript. To initialize the webserver you open the XAMPP Control Panel and start the Apache Webserver as well the MySQL database.

3.8.1 Webpage Development

The prototype has PHP files to enter data into the database, extract the data and a JavaScript to present the data into a map as the data is based on GPS coordinates. A script is run from the Arduino through the Wi-Fi module to write assumed data into the database which is a MySQL database. The database, tables and its configurations are as shown below:

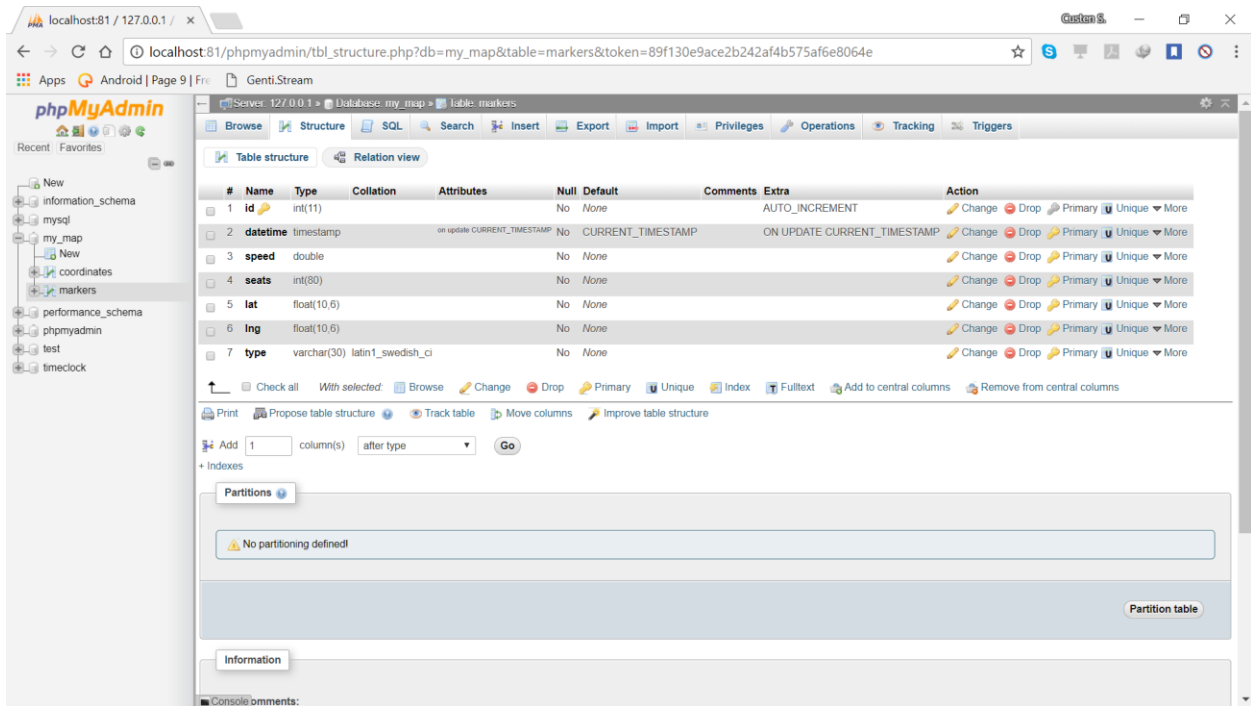
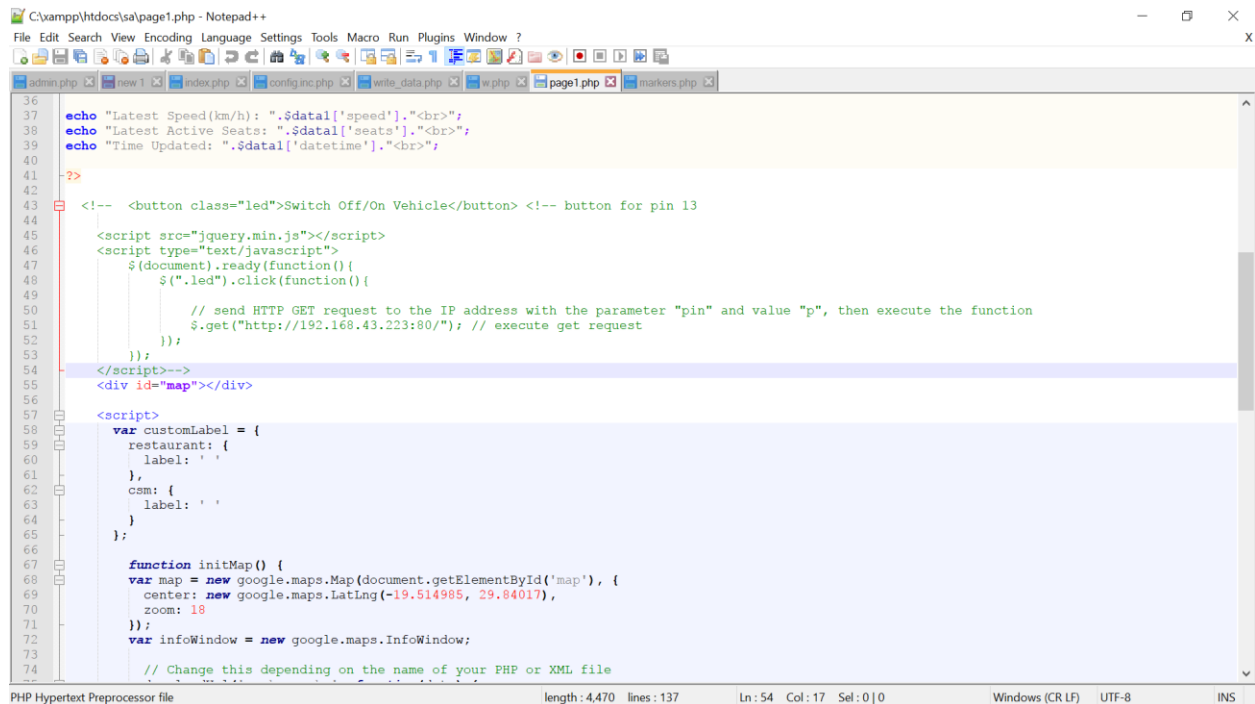


Fig 3.7 MySQL Database

The PHP pages are programmed using Notepad++, a free source code editor which supports a number programming languages that work under the Microsoft environment [15].



```
36 echo "Latest Speed(km/h): ".$data['speed']."<br>";
37 echo "Latest Active Seats: ".$data['seats']."<br>";
38 echo "Time Updated: ".$data['datetime']."<br>";
39
40
41 -?>
42
43 <!-- <button class="led">Switch Off/On Vehicle</button> <!-- button for pin 13
44
45 <script src="jquery.min.js"></script>
46 <script type="text/javascript">
47 $(document).ready(function() {
48 $(".led").click(function() {
49
50 // send HTTP GET request to the IP address with the parameter "pin" and value "p", then execute the function
51 $.get("http://192.168.43.223:80/"); // execute get request
52 });
53 });
54 </script>-->
55 <div id="map"></div>
56
57 <script>
58 var customLabel = {
59 restaurant: {
60 label: ''
61 },
62 csm: {
63 label: ''
64 }
65 };
66
67 function initMap() {
68 var map = new google.maps.Map(document.getElementById('map'), {
69 center: new google.maps.LatLng(-19.514985, 29.84017),
70 zoom: 18
71 });
72 var infoWindow = new google.maps.InfoWindow;
73
74 // Change this depending on the name of your PHP or XML file
```

Fig 3.8 Notepad++ editor

Notepad++ is a low-level developing tool. A webpage will sketch a map and plot the coordinates from the database and its URL is: localhost:81/sa/page1.php. This page will consistently refresh itself every 30 seconds so as to present the latest data received.

3.9 SYSTEM ARCHITECTURE

The system has a very active software side as the hardware. The combination of the two brings the completion of the prototype. The two subsystems are in constant communication so as to stay in sync. The entire system architecture is an Internet of Things endorser.

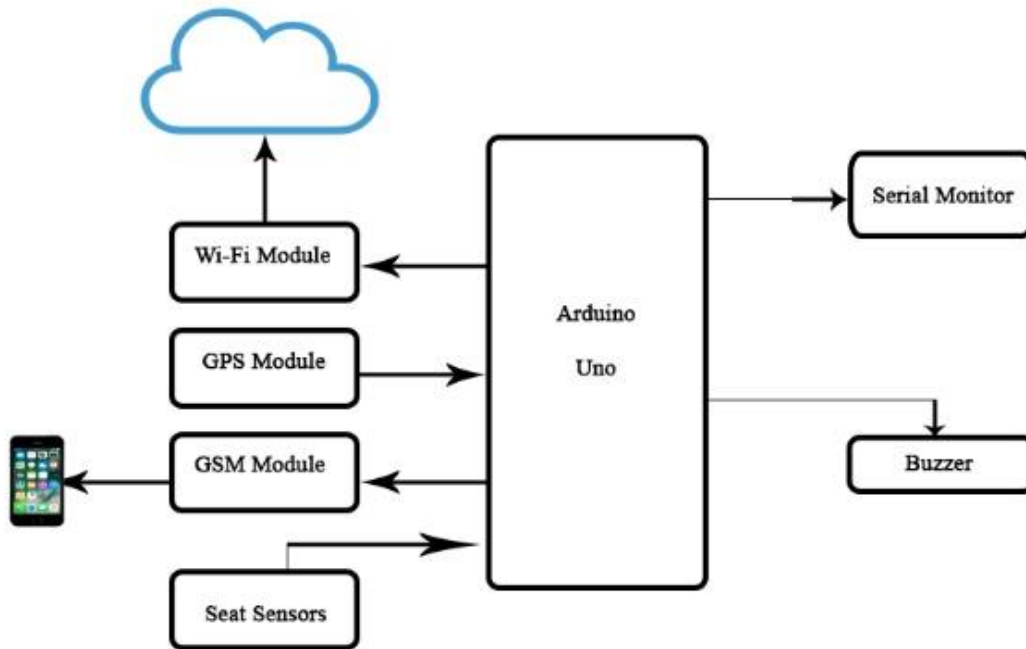


Fig 3.9 System Architecture

3.9.1 System Connection

The system components are connected to the Arduino as follows:

a) Power

- i) Arduino to Laptop via USB cable.
- ii) 12V AC-to-DC Adapter to GSM module.
- iii) Wi-Fi VCC pin to 3.3V Arduino potential
- iv) Wi-Fi CH_PD pin to 3.3V Arduino potential
- v) GPS VCC pin to 5V Arduino potential
- vi) Seat sensor positive plate to 5V Arduino potential
- vii) Buzzer positive pin to 5V Arduino potential
- viii) Push Button positive pin to 5V Arduino potential
- ix) Seat sensor resistor to Arduino ground
- x) Buzzer ground to Arduino ground
- xi) Push Button ground to Arduino ground

- xii) GSM module ground to Arduino ground
- xiii) GPS module ground to Arduino ground
- xiv) Wi-Fi module ground to Arduino ground

b) Serial Communication

- i) GSM module TXD to Arduino digital pin 2
- ii) GSM module RXD to Arduino digital pin 3
- iii) Wi-Fi module TXD to Arduino digital pin 4
- iv) Wi-Fi module RXD to Arduino digital pin 5
- v) GPS module TXD to Arduino digital pin 10
- vi) GPS module RXD to Arduino digital pin 11

c) Pin Read.

- i) Sensor read pins to Arduino digital pins 7,8 and 9
- ii) Push Button read pin to Arduino digital pin 12
- iii) Buzzer read pin to Arduino analog pin 4

3.10 SYSTEM FLOW CHART

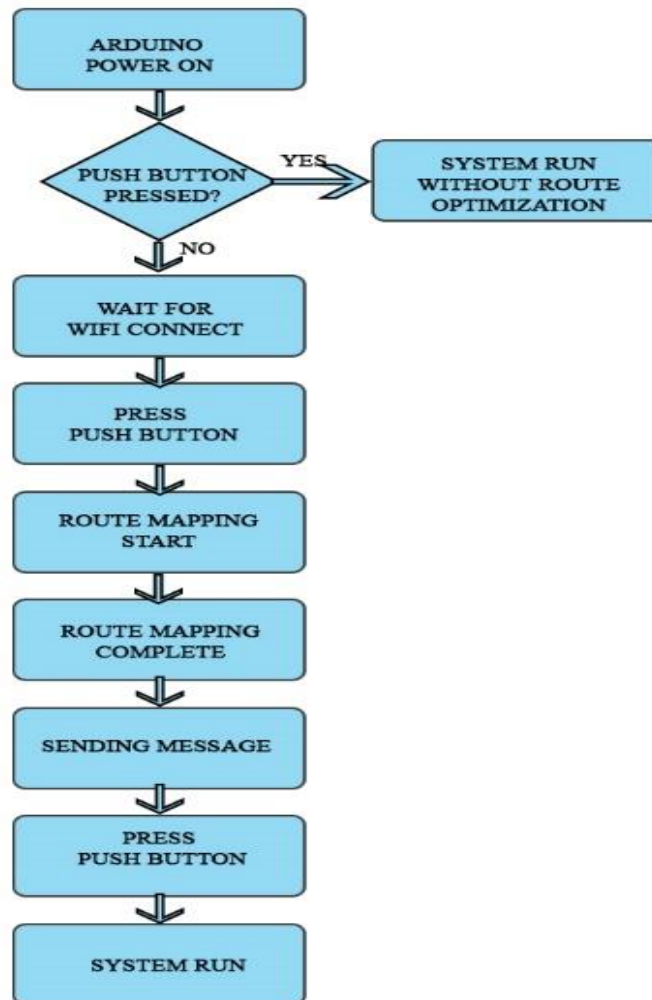


Fig 3.10 System flow chart

REFERENCES

40. Arduino. "Getting Started with the Arduino GSM Shield". Internet: <https://www.arduino.cc/en/Guide/ArduinoGSMShield/>, [Oct. 3, 2017].
41. Stack Exchange. "How to communicate the Arduino board with SIM900?" Internet: <https://arduino.stackexchange.com/questions/9483/how-to-communicate-the-arduino-board-with-sim900/>, [Oct. 3, 2017].
42. Stack Exchange. "Images." Internet: <https://i.stack.imgur.com/Pflu8.jpg>, [Oct. 2, 2017].
43. Open Source Library. "BETA_GSM_GPRS_GPS_IDE100_v307_1.zip" Internet: <http://www.gsm-lib.org/download.html>, [Oct. 2, 2017].
44. Instructables. "Arduino Ublox GPS." Internet: <http://www.instructables.com/id/Arduino-Ublox-GPS/>, [Oct. 2, 2017].
45. Waveshare. "UART GPS Module, NEO-7M-C onboard, straight/vertical pinheader." Internet: <https://www.waveshare.com/uart-gps-neo-7m-c-b.htm>, [Oct. 3, 2017].
46. Global Free Shipping. "Images." Internet: http://img.dxcdn.com/productimages/sku_312527_1.jpg, [Oct. 3, 2017].
47. Arduino. "Arduino Uno Image." Internet: https://store-cdn.arduino.cc/usa/catalog/product/cache/1/image/520x330/604a3538c15e081937dbfbd20aa60aad/A/0/A000066_featured_2.jpg, [Oct. 3, 2017].
48. GitHub. "A new, customizable Arduino NMEA parsing library." Internet: <https://github.com/mikalhart/TinyGPSPlus>, [Oct. 3, 2017].
49. ESP8266 WiFi Module Quick Start Guide. Kiwicon & Rancidbacon, 2016, pp. 1-20.
50. Room-15. "ESP8266 - AT Command Reference." Internet: <https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/>, [Oct. 3, 2017].
51. Electronics Hub. "Transistor as Switch." Internet: <http://www.electronicshub.org/transistor-as-switch/>, [Oct. 4, 2017].
52. School Physics. "The transistor." Internet: <http://www.schoolphysics.co.uk/age16-19/Electronics/Transistors/text/Transistor/index.html>, [Oct. 4, 2017].
53. Apache Friends. "XAMPP Apache + MariaDB + PHP + Perl." Internet: <https://www.apachefriends.org/index.html>, [Oct. 3, 2017].
54. Notepad++. "Notepad++ - About." Internet: <https://notepad-plus-plus.org/>, [Oct. 3, 2017].

CHAPTER 4

4.1 INTRODUCTION

This chapter serves to present and analyze the results found during a test of the entire system. Statistical data is shown in tables and coordinates represented on maps. An analysis of the results will be made and discussed explicitly. The results should be in line with the projects objectives.

4.2 THE PUBLIC TRANSPORT INVENTORY MONITORING SYSTEM

The system was prototyped to monitor and keep track of public transport vehicles. The system gets GPS coordinates and posts them into a database running on a local machine. The system has route optimization whereby it constantly checks if the received coordinates are in sync with those landmarked already. If the vehicle is found to be off route an alert is sent to the owner of the fleet after the third consecutive off-route coordinate is detected. The system has capabilities of counting the number of active seats within the vehicle.

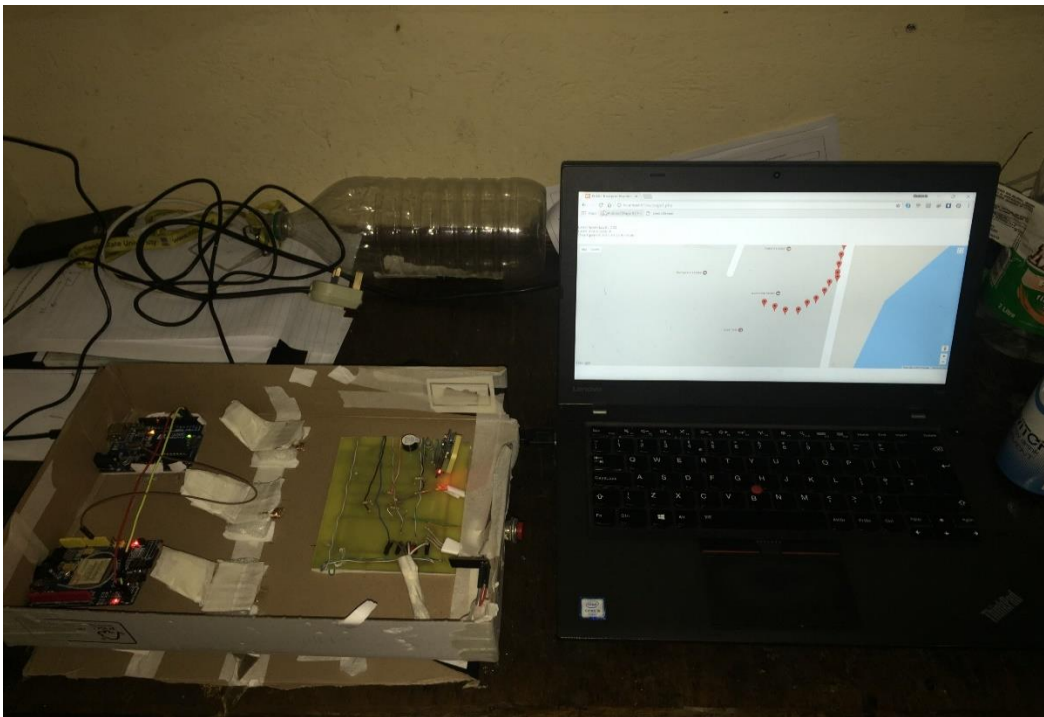


Fig 4.1 Overall System

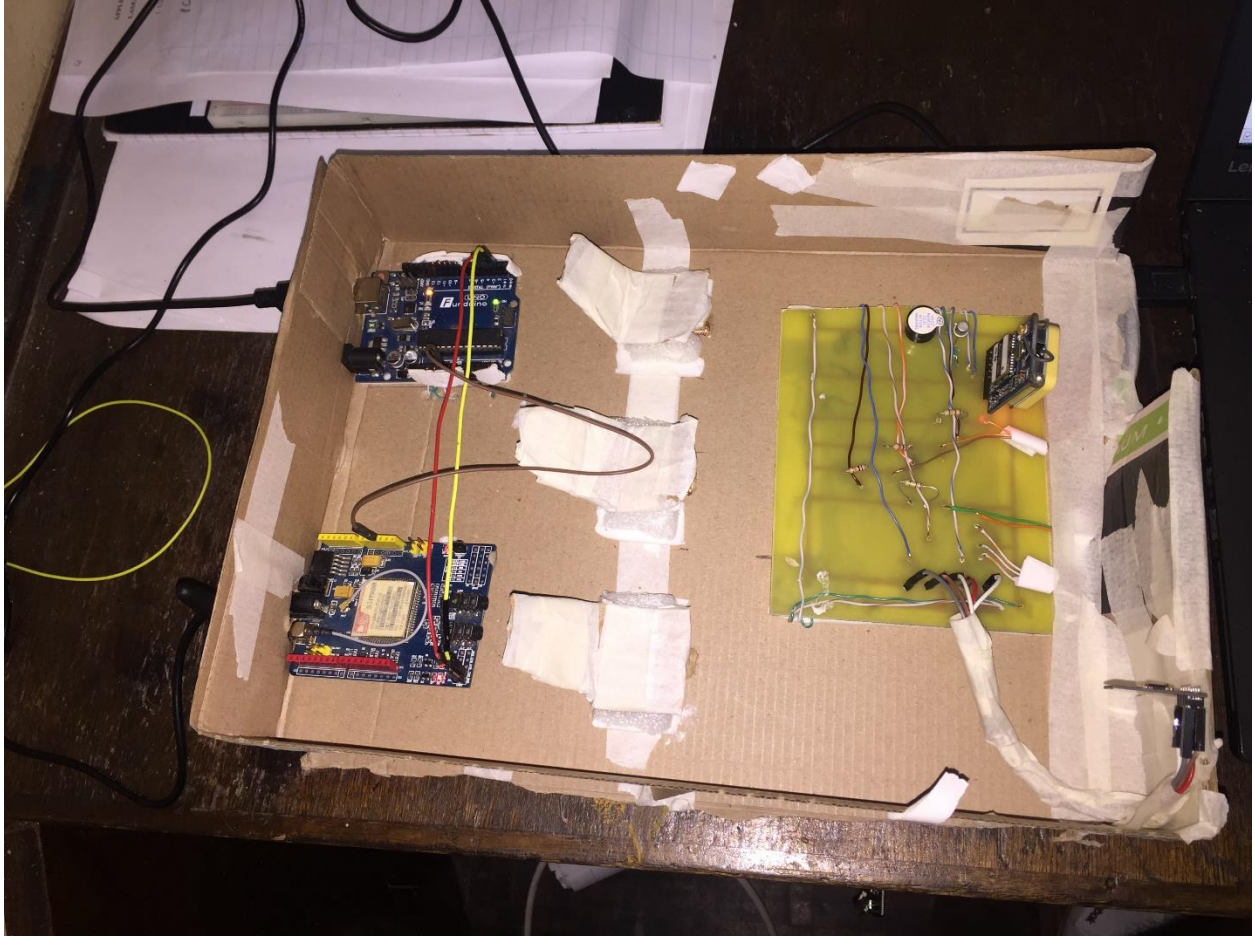


Fig 4.2 Circuit Image

4.3 WEBSERVER USER ACCESS PAGE

The owner of the fleet can view a map keeping track of the vehicle being tracked and also the various speeds and active seats at any given point.

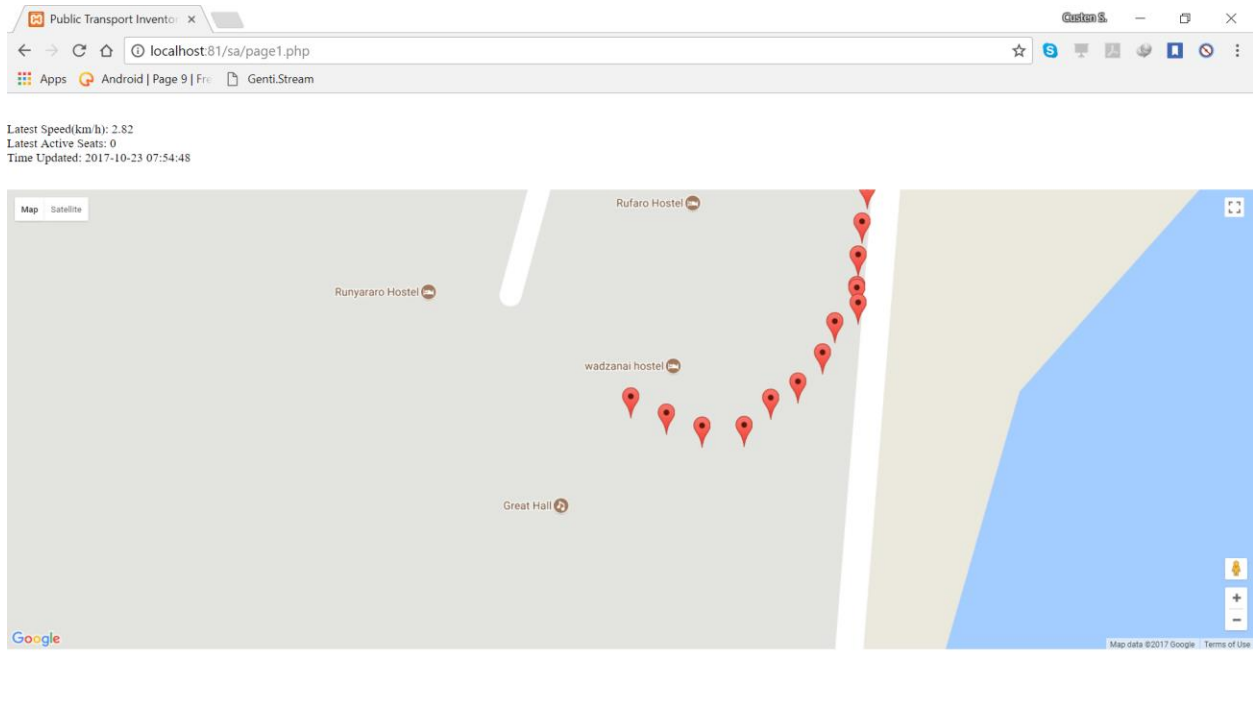


Fig 4.3 Webservice Access page

4.4 ROUTE MAPPING

As the system begins it takes the coordinates of 3 points with an interval of approximately 12 seconds. These three coordinates are then used in comparison with any coordinates received afterwards for route optimization. The route optimization comparison degree is corrected to the 4th decimal place of both the latitude and longitude.

4.4.1 Landmarks

Table showing 3 landmarks

Coordinate No.	Time(seconds)	Latitude	Longitude
1	15.39	-19.514904	29.840724
2	12.70	-19.514738	29.840738
3	13.07	-19.514595	29.840745

Table 4.1 Landmarks

Map showing landmarks

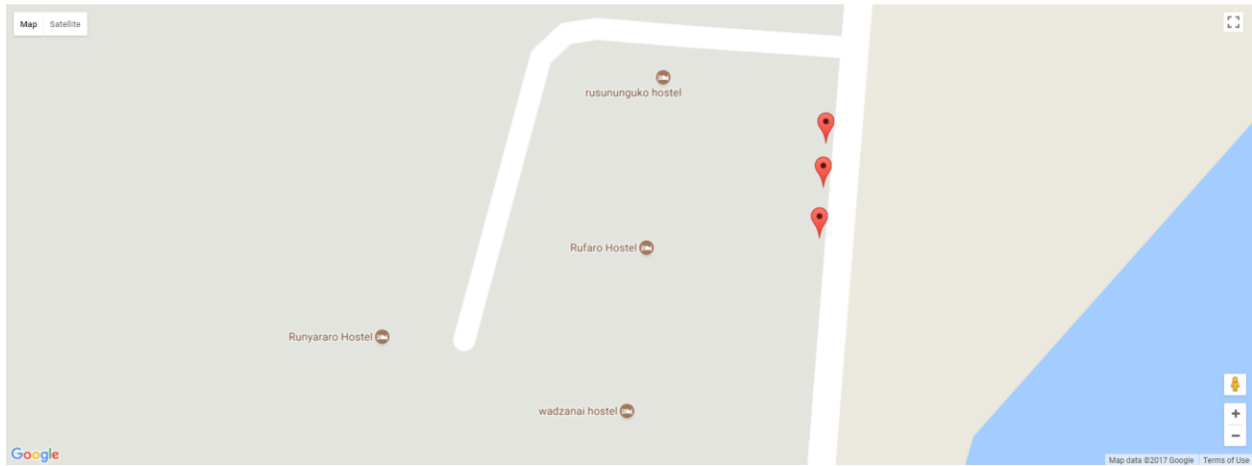


Fig 4.4 Landmarks Map

4.4.2 Route Optimization

The comparison of the landmarked coordinates and the newly received works very well. The system displays an “On Route” message on the serial monitor when the coordinate is on route. If the coordinate is off route it will increment a variable, display and send a text message when the variable has been consecutively incremented twice. If the next coordinate is on route the variable is reduced to zero.

```
Sending
AT+CIPSEND=87
Done

Latitude:
-19.515066
Longitude:
29.840709
Speed KPH:
1.93

On-route
Active seats: 1
Sending
AT+CIPSEND=87
Done

Latitude:
-19.515172
Longitude:
29.840698
Speed KPH:
2.17

Off route detected, count: 1
Active seats: 0
Sending
AT+CIPSEND=87
Done

 Autoscroll
Both NL & CR 9600 baud Clear output
Arduino/Genuino Uno on COM3
```

Fig 4.5 Serial Monitor

4.4.3 On-Route Coordinates

The test first shows the on route coordinates.

Table showing On-Route Coordinates

Date and Time	Speed(km/h)	Active Seats	Latitude	Longitude
2017-10-23 07:51:20	1.93	1	-19.5151	29.84071
2017-10-23 07:51:03	1.54	2	-19.515	29.84073
2017-10-23 07:50:46	1.48	1	-19.5149	29.84073
2017-10-23 07:50:29	2.63	0	-19.5148	29.84073
2017-10-23 07:50:12	2	0	-19.5147	29.84074
2017-10-23 07:49:55	1.85	0	-19.5146	29.84075
2017-10-23 07:49:37	0.07	0	-19.5145	29.84074
2017-10-23 07:49:21	0.43	0	-19.5145	29.84075
2017-10-23 07:49:04	0.15	0	-19.5146	29.84073
2017-10-23 07:48:47	0.04	0	-19.5146	29.84073
2017-10-23 07:48:30	0.07	0	-19.5146	29.84074
2017-10-23 07:48:13	0.07	0	-19.5145	29.84074
2017-10-23 07:47:56	4.06	0	-19.5146	29.84075

Table 4.3

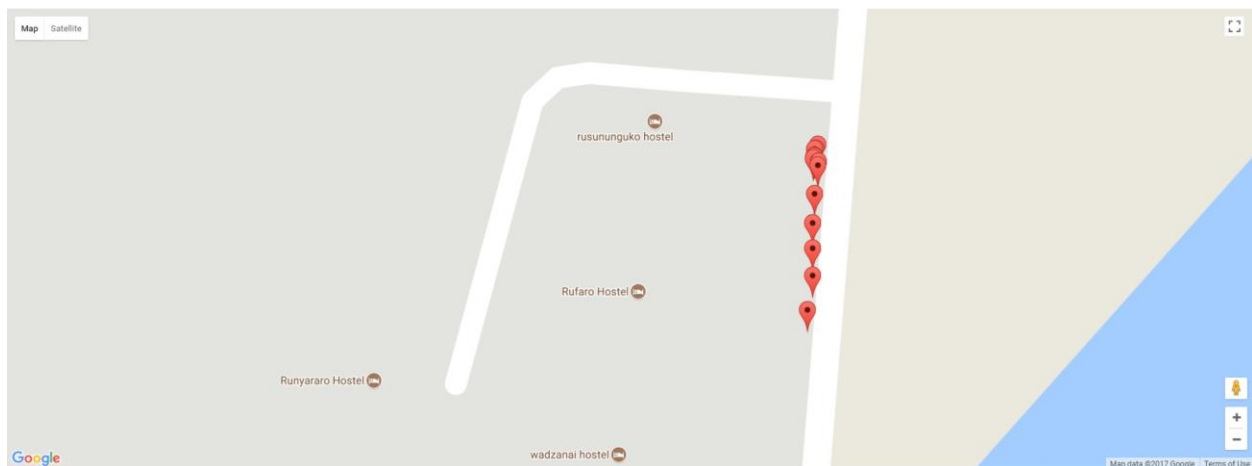


Fig 4.5 On-route coordinates map

4.4.3 Off-Route Coordinates

Table showing Off-Route Coordinates

Date and Time	Speed(km/h)	Active Seats	Latitude	Longitude
2017-10-23 07:54:48	2.82	0	-19.5156	29.83991
2017-10-23 07:54:30	1.85	0	-19.5157	29.84004
2017-10-23 07:54:13	1.67	0	-19.5157	29.84016
2017-10-23 07:53:56	2.91	0	-19.5157	29.8403
2017-10-23 07:53:38	2	0	-19.5156	29.8404
2017-10-23 07:53:21	2.2	0	-19.5156	29.84049
2017-10-23 07:53:04	3.13	0	-19.5155	29.84057
2017-10-23 07:52:46	1.87	0	-19.5154	29.84062
2017-10-23 07:52:29	2.93	0	-19.5153	29.8407
2017-10-23 07:52:12	0.19	0	-19.5153	29.84069
2017-10-23 07:51:53	0.11	0	-19.5153	29.84069
2017-10-23 07:51:36	2.17	0	-19.5152	29.8407

Table 4.3

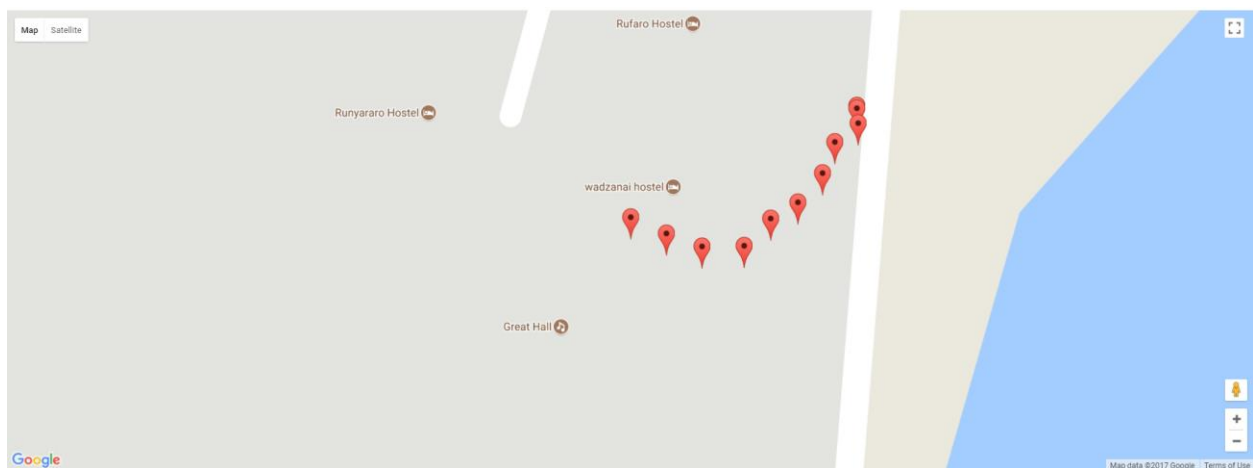


Fig 4.6 Off-route coordinates map

4.5 SMS ALERTS

The system sends a message to the owner of the fleet when there are three consecutive off routes detected. Below is a table showing the time taken for an SMS to be delivered.

SMS Alert Latencies

Test Number	Time Taken (seconds)
1	12.63
2	9.07
3	11.49
4	14.87
5	16.13

Average time taken is 12.84 seconds.

Table 4.4

4.6 ANALYSIS

1. Power connections to modules are very delicate as jumper wires were used.
2. SMS alerts take more time in initializing the text before sending it, this is because of the GSM library used in the code.
3. The GPS module is relatively accurate, though with a few unprecise values popping up once in a while.
4. The webpage may take time to display all the required information because it is using the Google API which needs internet access, thus the quick reload of the page depends on the internet connectivity.
5. The Wi-Fi module is very fast in posting the data to the database thus making new information be available in real time.
6. Some stability issues occur with the Arduino as the code exceeds the recommended Dynamic RAM usage.

CHAPTER 5

5.1 INTRODUCTION

The designing and operation of the system was a success though with hiccups here and there. It was a learning curve for the researcher to then make the system to meet its objectives and troubleshoot shortfalls. The system gives the required results with sizeable accuracy.

5.2 SYSTEM DRAWBACKS

The system has a few drawbacks which include:

- The system cannot identify if an active seat has been activated by a human or luggage.
- The system may not differentiate if an active is sat on by a paid passenger or it's the omnibus staff.
- The system uses a Virtual Private Network (VPN) which might be a problem to connect to in very remote areas.

5.3 CHALLENGES FACED

During the development of the system challenges were faced but the researcher managed to emerge victorious. These challenges include:

- The SIM900 GSM module used in this project had some functionality issues thus it could not be powered by the Arduino hence not making the prototype completely mobile.
- The researcher has done something that a very few population has managed to do that is to have 3 different modules on one circuit, the issue with having this is that the serial communications between the Arduino and one of the modules may be interfered by the other module.

5.4 RECOMMENDATIONS

The researcher recommends some areas of further research which include:

- Installing cameras within the vehicle to monitor the activity and curb the drawback of active seat differentiation.
- To monitor the functional state of the vehicle that is fuel levels, oil levels, engine temperature, etc.

5.5 CONCLUSION

In conclusion the research project was a success and the prototype meets all its objectives. The aim of the research was to develop a system that will track the movement of a member of a fleet, enforce route optimization and know how many passengers are aboard the trip all in real-time. This was achieved and the system that meets that aim exists.

APPENDIX

```
#include "TinyGPS++.h"
#include "SIM900.h"
#include "SoftwareSerial.h"
#include "sms.h"

SoftwareSerial serial_connection(10, 11); //RX=pin 3, TX=pin 2
SoftwareSerial client(4, 5);

TinyGPSPlus gps;//This is the GPS object that will pretty much do all the grunt work with the
NMEA data

double coordlat[3];
double coordlng[3];
int i=0;

void wifi_init()
{
client.begin(115200);
client.flush();

    client.println("AT");
    Serial.println("AT");
    delay(2000);
    if(client.find("OK")) {
Serial.println("OK");
    }

    Serial.println("Connecting Wifi..");

    client.println("AT+CWJAP=\"Custen's iPhone\", \"Passw0rd\""); //provide your WiFi
username and password here
    delay(10000);
client.end();

    if(i==0){
        Serial.println("Connected,waiting for sys start");
        while(digitalRead(12)==0){
            }}

    Serial.println("System Start");
}
```

```

void sendData(int seat){
  serial_connection.end();
  client.begin(115200);
  client.flush();
  String cmd = "AT+CIPSTART=\"TCP\", \"";          // Establishing TCP connection
  cmd += "172.20.10.3";                          // api.thingspeak.com
  cmd += "\",81";
  client.println(cmd);
  if(client.find("ERROR")) {
    Serial.println("AT+CIPSTART error");
  }
  else {
    Serial.println("Sending");
  }
  String getStr = "GET /sa/w.php?lt=";           // prepare GET string
  getStr += String(gps.location.lat(),6);       // Latitude Data
  getStr += "&lg=";
  getStr += String(gps.location.lng(),6);       // Longitude Data
  getStr += "&sd=";
  getStr += String(gps.speed.mph()*1.60934);
  getStr += "&st=";
  getStr += String(seat);
  getStr += "HTTP/1.1\r\nHost: 172.20.10.3:81\r\n\r\n";
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());              // Total Length of data
  client.println(cmd);
  Serial.println(cmd);
  client.println(getStr);
  Serial.println("Done");
  delay(5000);
  client.println("AT+CIPCLOSE");              // closing connection
  client.end();
  serial_connection.begin(9600);
}

```

```

}

double trunc(double d)
{
    return (d>0) ? floor(d) : ceil(d) ;
}

void newfunc(){
    while(serial_connection.available())//While there are characters to come from the GPS
    {
gps.encode(serial_connection.read());//This feeds the serial NMEA data into the library one
char at a time
    }

    if(gps.location.isUpdated())//This will pretty much be fired all the time anyway but will at
least reduce it to only after a package of NMEA data comes in
    {
        if(i==0) {
analogWrite(A4,500);
delay(300);
analogWrite(A4,0);

        //Serial.println("Route Mapping Starting, positions will be noted thrice after every
15seconds, starting now.");

        delay(15000);
    }

    Serial.print("Updating coordinate No: "); Serial.println(i+1);
    coordlat[i] = (gps.location.lat());
    coordlng[i] = (gps.location.lng()); //delay(10000);
    analogWrite(A4,500);
    delay(300);
analogWrite(A4,0);
if(i<3) {
    Serial.println("Done!next position");
    delay(2000);
    }
    if(i==2){

```

```

        Serial.println("Route Mapping Complete.");
Serial.println("Sending message...");
    gsm.begin(9600);
    SMSGSM sms;
    sms.SendSMS("0783235284", "Route Mapping Complete.");
analogWrite(A4,500);
delay(700);
analogWrite(A4,0);
    Serial.println("waiting");
    while(digitalRead(12)==0){
        }
        analogWrite(A4,500);
    delay(300);
analogWrite(A4,0);
    for(int j=0; j<3; j++){
        Serial.print("Lat: ");Serial.println(j+1);
        Serial.println(coordlat[j], 6);
        Serial.print("Lng: ");Serial.println(j+1);
        Serial.println(coordlng[j], 6);
    }
}
i++;
delay(10000);
}
}

void optimiz(){
    bool enroutelat, enroutelng , routelat[3], routelng[3];
for(int x=0; x<3; x++){
    if (trunc(1000. * gps.location.lat()) == trunc(1000. * coordlat[x])) {
        routelat[x]=true;
    }
    else{

```



```

    routelat[x]=false;
}
}
for(int x=0; x<3; x++){
    if (trunc(10000. * gps.location.lng()) == trunc(10000. * coordlng[x])) {
        routelng[x]=true;
    }
    else
        routelng[x]=false;
}
if(routelat[0]==true or routelat[1]==true or routelat[2]==true){
    enroutelat=true;
}
if(routelng[0]==true or routelng[1]==true or routelng[2]==true){
    enroutelng=true;
}
else{
    enroutelat=false;
    enroutelng=false;
}
    if(coordlat[0]!=0.00 && coordlng[0]!=0.00){
        if(enroutelat==false or enroutelng==false){
            i++;
Serial.print("Off route detected, count: ");
Serial.println(i-4);
        }
        else {
            Serial.println("On-route");
            i=4;
            analogWrite(A4, 0);
        }
        if(i==7) {
            analogWrite(A4, 500);

```

```

gsm.begin(9600);
  SMSGSM sms;
  Serial.println("Sending SMS");
  sms.SendSMS("0783235284", "Off Route Detected");
  i=4;
  }
}

void setup(){
  analogWrite(A4, 500);
  delay(300);
  analogWrite(A4, 0);
  if(digitalRead(12)==1){
    analogWrite(A4,500);
    delay(300);
  }
  analogWrite(A4,0);
  i=4;
}

  Serial.begin(9600);//This opens up communications to the Serial monitor in the Arduino IDE
  wifi_init();
  serial_connection.begin(9600);//This opens up communications to the GPS
}

void loop()
{
  while(serial_connection.available())//While there are characters to come from the GPS
  {
    gps.encode(serial_connection.read());//This feeds the serial NMEA data into the library
    one char at a time }
  if(gps.location.isUpdated())//This will pretty much be fired all the time anyway but will at
  //least reduce it to only after a package of NMEA data comes in
  {
    while(i<3){
      newfunc();
    }
  }
}

```

```

        if(i==3){
            i=4;
        }
    }

    //Get the latest info from the gps object which it derived from the data sent by the GPS unit
    Serial.println();
    Serial.println("Latitude:");
    Serial.println(gps.location.lat(), 6 );
    Serial.println("Longitude:");
    Serial.println(gps.location.lng(), 6);
    Serial.println("Speed KPH:");
    Serial.println(gps.speed.mph()*1.60934);
    Serial.println("");
    optimiz();
int seat=0;
    if (digitalRead(7)==1) {
        seat++;
    }
    if (digitalRead(8)==1) {
        seat++;
    }
    if (digitalRead(9)==1) {
        seat++;
    }
    Serial.print("Active seats: ");
    Serial.println(seat);
    sendData(seat);
    seat=0;
    Serial.println("");
delay(10000);
}
}

```