College of Engineering

Mechanical Engineering Department

Information Technology and Computer Engineering Department

Develop a drive assistant system to improve dynamic stability and safety of vehicle

Project Team

Rami Awiwi Salem Najajrah

Khaleel Abu Turki Awfa Sarahneh

Supervisor

Dr. Momen Sughayyer Dr. Hashem Tamimi

Submitted to the College of Engineering and Computer Engineering
in partial fulfillment of the requirements for the
Bachelor degree in Automotive Engineering
Palestine Polytechnic University

2022

Hebron-Palestine

Dedication (Arabic)

إلى من تعهداني بالتربية في الصغر ، وكانا لي نبراساً يضيء فكري بالنصح ، و التوجيه في الكبر أمي ، وأبي

حفظهما الله

إلى من شملوني بالعطف ، وأمدوني بالعون ، وحفزوني للتقدم ، إخوتي ، وأخواتي

رعاهم الله

إلى كل من علمني حرفاً ، وأخذ بيدي في سبيل تحصيل العلم ، والمعرفة

إليهم جميعاً أهدي ثمرة جهدي ، ونتاج بحثي المتواضع

فريق المشروع

Abstract

The general objective of the project is improve safety for passenger during travel by develop a system based on guiding the driver to determine the appropriate speed and steering angle based on the diminution of the road and the specification of the vehicle such as (weight, distance between the two wheels...etc.), and the actual vehicle information such as speed, wheel steering angle and vehicle coordinates collected. It enters into calculations in order to achieve the main goal of avoiding loss control of vehicle accident and deviating from the lane.

In this project, we relied on several auxiliary technologies, such as Global Positioning System **GPS**, which helps to know the actual location and coordinates of the vehicle, and the Geographic Information System **GIS**, which helps to know general information about the road such as turns, road length and width, and finally Electronic Control Unit **ECU**, where extract the information from the electronic control unit such as speed and steering angle, and finally using the vehicle dynamics equations that work to calculate the appropriate speed of the vehicle and the appropriate angle of turning based on the previous inputs. The simulation process was carried out by MATLAB program. In this project, the actual reading was taken from the electronic control unit in the vehicle and the location of the vehicle. and linked in MATLAB

After the experiment and make many sample of same rod (inside the university), take 190 value for vehicle speed, steering angle, longitude and latitudes data, then after Data processing by matlab the simulation result for vehicle speed and steering angle command is very reliable .

الملخص

الهدف العام للمشروع هو تطوير نظام يقوم على إرشاد و توجيه السائق ومساعدته على تحديد السرعة المناسبة وزاوية التوجيه بناءً على المعلومات الهندسية للطريق وخصائص المركبة مثل (الوزن ، المسافة بين العجلين ...إلخ) ، ومعلومات المركبة الفعلية مثل سرعة المركبة وزاوية توجيه العجلات واحداثيات المركبة و جميعها تدخل في عمليات الحسابية من أجل تحقيق الهدف الرأيسي الا وهو تجنب الحوادث والانحراف عن المسار.

في هذا المشروع ، اعتمدنا على العديد من التقنيات المساعدة ، مثل **GPS** ، والتي تساعد في معرفة الموقع الفعلي واحداثيات المركبة ، و نظم المعلومات الجغرافية **GIS** التي تساعد على معرفة المعلومات العامة عن الطريق مثل المنعطفات وطول الطريق وعرضه وأخيرا وحدة التحكم الإلكترونية **ECU**، حيث نقوم باستخراج المعلومات من وحدة التحكم الإلكترونية مثل السرعة و زاوية التوجيه ، وأخيراً باستخدام معادلات ديناميكا المركبات التي تعمل على حساب السرعة المناسبة للسيارة وزاوية الانعطاف المناسبة بناءً على المدخلات السابقة يتم إجراء عملية المحاكاة بواسطة برنامج MATLAB في هذا المشروع تم أخذ القراءة الفعلية من وحدة التحكم الإلكترونية في السيارة وموقع السيارة وربطها فيMATLAB

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abbreviations

GPS: global positioning system.
ADAS: Advanced Driver Assistance Systems.
ACC: Adaptive Cruise Control.
FCW: Forward Collision Warning**.**
ISA: Intelligent Speed Assistance**.**
LWD: Lane Departure Warning.
LKS: Lane Keeping System**.**
LCA: Lane Change Assistance.
SA: Selective Availability.
GIS: Geographic information system.
ECE:United Nations Economic Commission for Europe.
WHO:World Health Organization.
VSA: Vehicle Stability Assist.
VDC: Vehicle Dynamic Control.
VSC: Vehicle Stability Control.
ESP: Electronic Stability Program.
ESC: Electronic Stability Control.
DYC: Direct Yaw Control.
iBooster:Electromechanical Brake Booster.
IDE: integrated development environment
PLC: and programmable logic controls
PIC: Programmable Intelligent Computer
Kb: kilo bite
PC: personal computer
mm: millimeter
ms. : millisecond
ADC: analog to digital converter
ECU: electronic control module
LED: Light-emitting diode

List of symbols

NO	Symbols	Description
1	c.g	Center of Gravity
2	L	Wheel Base
3	δ	Steering Angle
4	ψ	Orientation of the Vehicle
5	V	The velocity at the c.g of the vehicle
6	β	Slip Angle
7	m	Mass of the Vehicle
8	a_y	Lateral Acceleration
9	F_{yf} and F_{yr}	Lateral Tire Forces of the Front and Rear Wheels
10	$V_x\dot{\psi}$	Centripetal Acceleration
11	α_f and α_r	Slip Angle of the Front and Rear Wheel
12	$\theta_{vf}$and θ_{vr}	Is the Angle that the Velocity Vector Makes with the Longitudinal Axis of the Vehicle
13	$C\alpha_f$ and $C\alpha_r$	Front and Rear Cornering Stiffness
14	$\dot{\psi}$	yaw rate of vehicle body
15	I_z	yaw moment of inertia

1

CHAPTER 1

Introduction

This chapter provides an introduction to the project. It starts with a motivational statement, followed by the aims and objectives, a brief description, a discussion of existing work and finally the scope and constraints of the project.

1.1 Overview

As the worldwide use of automobiles increases rapidly, it has become even more important to develop vehicles that optimize the use of road and fuel resources, provide safe and comfortable transportation and at the same time have minimal impact on the environment. To meet these diverse and often conflicting requirements, automobiles are increasingly relying on electronics systems that employ sensors, actuators and feedback control.

Due to a variety of factors, private transportation is becoming more prevalent in Palestine. As a result, Palestinians are increasingly spending in securing their own vehicles. As a result, modern vehicles may be seen circulating on Palestinian roadways, implying that cutting-edge automotive technology is becoming more readily available. Driver assistance systems, car orientation and navigation systems, and other technologies are now available and require the attention of highly skilled and experienced technicians and engineers, which is a new trend for the local target industry.

Any system that can provide intelligent vehicle location and navigation information, with connection to ECU, that help us to ensure our safe drive, saving people life and cars. The vehicle assistance systems have been developed to obtain an optimal and safe driving experience. These systems have been developed using software that rely primarily on different sensors that increase the accuracy of the results and so better response.

1.2 Motivation

The concept of self-driving cars has become increasingly popular. The automotive industry is investing heavily in the development of self-driving systems. This interest is largely due to the promising features and characteristics of modern self-driving systems. To begin with, studies estimate that approximately 1.35 million people die in traffic accidents in 2020 **[14]**, most of which are mainly caused by human factors. The argument is, even though many of today's cars are offered with top-of-the-line passive safety features like seatbelts and airbags, it is still not as good as the active safety features of self-driving cars. These active features can help predict and avoid accidents before they even happen. Additionally, the autonomous and precise nature of self-driving cars can also help reduce fuel consumption and drive down emissions caused by road vehicles.

The importance and promising future of this technology have led to a great deal of research. In order to establish a functioning self-driving system, two main tasks must be carried out. To start with, the self-driving car must be capable of understanding its surrounding environment as well as realizing its position and orientation relative to that environment. The second, and perhaps the most distinguishable task, is mainly concerned with how to react properly to the detected

surroundings while progressing on route. This is done through years of research, testing and development.

Machine learning techniques used in self-driving systems have to undergo a training process. In supervised learning, the system is supplied with labeled training data. The labels indicate how the system should react to certain situations. However, the use of modern technologies such as GPS and GIS, such as what we used in this project, contribute to obtaining optimal driving, improving road use, saving fuel and obtaining safe driving.

Using GPS in vehicle dynamic is based on a set of main inputs “in a simplified way” such as the location of the vehicle and the road information “a lane” such as the permissible speed, the ideal speed in this lane or curve and the distance of the road and linking all this information to warn by display to type alarm speed and steering angle recommended in a timely manner.

1.3 Problem statement

Every day 3700 people die due to accidents that occur on the road, either due to the shortcomings of one or both parties due to non-compliance with traffic laws or because of distraction, either due to work fatigue or the use of a smartphone while driving and Consequence Every year, approximately 1.35 million people die as a result of vehicle accidents **[14].** An additional 20 million to 50 million people suffer non-fatal injuries and many become disabled as a result. Injuries caused by traffic accidents cause great economic losses to individuals, their families and entire nations. These losses arise from the cost of treatment and lost productivity for people who die or become disabled due to their injuries, and family members who are forced to miss work or school to care for the injured. Traffic accidents in most countries cost 3% of GDP. According to what was announced by the World Health Organization in June 2021. **[14]**

In real time the main objective of most car assistance systems is to provide safety for the driver and passengers while driving. With integration between GPS, GIS and some data from ECU in the vehicle with software have mathematical vehicle model to make process of data to send alarm to driver while driving then we can achieve better and safe driving.

1.4 Aims and Objectives

The goal of this project is to develop a module to improving safety for driver and passenger as a driving assistance system, that are based on GPS, GIS, and some information from the vehicle's ECU, to process it using a dynamic mathematical model.

Specific objectives:

- Transfer data from the GPS receiver to the GIS, specifying the car's location and direction.
- Transfer data from GIS to the model, specifying the road data
- Transfer data from ECU of the vehicle to the model.
- Making simulation of the project.
- Warning the driver in a timely manner can reduce the occurrence of accidents due to improper speed of the road by determine the safe speed at real time.
- Preforming road tests.

1.5 Importance

Improving the quality of vehicle driving, which is reflected in the security and stability of the vehicle, and the full control of the vehicle, which also helps in road management, as well as increasing the accuracy of control in self-driving cars.

1.6 Methodology

This project links a vehicle dynamics system and a road navigation system together, developing a system that includes a GPS sensor (spark fun) work by Arduino UNO receiver, GIS map, microcontroller (Arduino Uno), and a processor that will be a personal respect to laptop computer CPU with MATLAB as a programming tool to achieve the prototype goals. When a target control system is adopted and implemented as a final product system in automobiles, it will, of course, help reduce accidents and save lives while also promoting autonomous vehicle research.

To complete this project, a multidisciplinary team comprised of mechanical and computer system engineers will collaborate and work together to achieve the goal of the project**.**

1.7 Project requirements

- **Hardware requirements:** Microcontroller (Arduino Uno) kit, GPS receiver kit, OBD (II) adapter and personal laptop, MCP2515 CAN Bus, Real Vehicle for test.

- **Software requirements:** Arduino software, (Arc Map) GIS Software**,** MATLAB program.

Explain the components in the chapter three and specify the details of each component.

1.8 Action plan

Over the course of two semesters, we will focus on creating the simulation process, and conducting hands-on experiments. The first stage consists of starting to define the idea and collecting data as shown in **Tables 1.1.**

❖ **First semester**

- **Stage one:** Identifying the project idea.
- **Stage two:** Project requirement and collecting data.
- **Stage three:** Modeling and calculation.
- **Stage four:** Writing and documentation.
- **Stage five:** Simulation using the MATLAB program

In the second semester, evaluated in extracting data and experimenting with electronic cutting and connecting shown in **Tables 1.2.**

❖ **Second semester**

- **Stage one:** Extract data from ECU
- **Stage two:** Preparing GPS and GIS data
- **Stage three:** Connection and interfacing of systems
- **Stage four:** Testing the Module

Table1.1. .Action plan for the first semester

Task\Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Identifying the project idea.																
project requirement and collecting data																
Modeling and calculation																
Writing and documentation.																

Table1.2. Action plan for the second semester

Task\Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Extract data from ECU																
Preparing GPS and GIS data																
Connection and interfacing of systems																
Testing the Module																

2

Chapter 2

Background

This chapter explores the different theoretical aspects of the project. It provides a clear and brief description of the techniques employed in this project.

2.1. Global Positioning System (GPS)

2.2.1 Introduction

GPS technology is a promising technology that has applications in many aspects of life, such as in agriculture, law, sports, the automobile industry, etc. In the automobile industry, GPS is used in many forms. Any system capable of providing intelligent navigation and location information of a car, that will help us to avoid obstacles, crashes and drive safely. A Global Positioning System provides the answer to facilitate all these issues.

The purpose of GPS in this project is to build and develop an algorithm that takes in inputs from a GPS receiver and using those inputs to successfully navigate a car through a set of known points (from GIS) and called it as waypoints. The entire process is divided into three parts. In the first part, the car is placed at the starting waypoint and the GPS receiver modifies the visible GPS satellites and tries to calculate the current latitude and longitude. In the second part, the microcontroller algorithm calculates the direction of the next waypoint from the current waypoint. In the third step, the algorithm calculates the distance to the next waypoint to determine what the driver should do.

2.2.2 How Car GPS works

GPS-based autonomous navigation is a very rapidly evolving technology. Researchers have developed several techniques for navigating between different external environments. This system is widely used in land vehicle navigation applications. The main advantage of using GPS is the data collected does not depend on previous readings, so localization errors do not grow over time. The downside is its precision. This depends on the environment and the number of satellites read. Generally, the problem of positioning (localization) the car consists of answering the question where am I?

During continuous motion, each satellite in the GPS constellation moves continuously sends radio signals in all directions. This information contains data about its trajectory. Device status and exact time. Car GPS receiver consists of an antenna and a computer with a screen. The receiver computer has a digital road map in memory (in some cases, on a CD-ROM) that contains most roads in the country. Thus, all you need to do is just put in the details of your destination on the computer screen and then wait for the details the computer is going to give you. **Figure 2.1**

Figure 2.1 Components of the car GPS system [2]

The computer clock and the satellite clocks are synchronized. Since the center of gravity of the earth influences the rate at which the clocks change, the clocks are not synchronized with respect to each other but as a function of their velocity with respect to the center of the coordinate system, which is the center of the earth. Packets from a satellite A contain a time stamp t_1 as part of the packets. Upon receiving these packets, the receiver clock reads its time t_2.

The frequency of transmission is f and the wavelength of transmitted radio signals is λ. **[2]** As a result of synchronization of the clocks, the computer determines the distance between the satellite and the car as follows:

The one-way delay of radio signals is: $t = t_2 - t_1$

The speed of radio signals is: $v = f\lambda$

The distance between satellite and car is: speed * delay = $f\lambda\ (t_2 - t_1)$.

2.2.3 How accurate is GPS?

There are several different levels of accuracy that can be achieved. The difference is equipment and techniques. In the commercial world, there have been roughly 4 generations of equipment, with some from the last three still in service.

Levels of accuracy provided by GPS.

1. **Standard Positioning Service (SPS):** is a positioning and timing service, and available to all GPS users. Provides the lowest accuracy GPS position measurements, normally in the region of 5-10 m.

2. **Precise Positioning Service (PPS):** is a highly accurate military positioning, velocity and timing service. Provides the accuracy GPS position measurements, normally in the region of 2-9 m.

3. **Code-Phase differential GPS or DGPS:** is an enhancement to the Global Positioning System (GPS) which provides improved location accuracy. Provides the accuracy GPS measurements, normally in the region of 1-5 m.

4. **Carrier-Phase differential GPS or CDGPS:** is a much more accurate Global Positioning System Provides the accuracy in sub-meter. **[4]**

2.2.4 Sources of GPS Errors

GPS pseudo range measurements are affected by several types of random errors and biases (systematic errors). These errors may be classified as those originating at the satellites, those originating at the receiver, and those that are due to signal propagation (atmospheric refraction). **5]**

1. **Errors from satellites include:**
   - Ephemeris
   - Orbital errors
   - Satellite clock errors, and
   - The effect of selective availability.

2. **Errors from Receiver include:**
   - Receiver clock errors
   - Multipath error
   - Receiver noise, and
   - Antenna phase centre variations.

3. **Errors from signal propagation include:**
   - The delays of the GPS signal as it passes through the ionospheric and tropospheric layers of the atmosphere.
   - In fact, it is only in a vacuum (free space) that the GPS signal travels, or propagates, at the speed of light.

In addition to the effect of these errors, the accuracy of the computed GPS position is also affected by the geometric locations of the GPS satellites as seen by the receiver. The more spread out the satellites are in the sky, the better the obtained accuracy **Figure 2.3**.

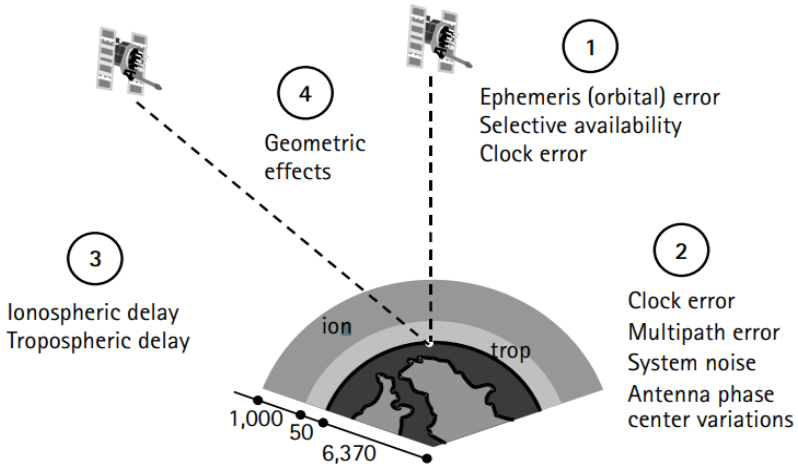


Figure 2.3. GPS errors and biases.

2.2.5 Geographical Information Systems GIS

is expressed in individual letters **G – I – S** and not at pronunciation GIS. It stands for Geographic or Geographical Information Systems. Geographic Information Science is an interdisciplinary field. It is built upon knowledge from geography, cartography, computer science, mathematics etc.

GIS can be defined as 'A system for Capturing, storing, checking, integrating, manipulating, analyzing and displaying data which are spatially referenced to the Earth. This is normally considered to involve a spatially referenced computer database and appropriate applications software'. GIS needs spatial data; this makes it unique. **[6]**

Each of these separate thematic maps is referred to as a **layer, coverage, or level**. And each layer has been carefully overlaid on the others so that every location is precisely matched to its

corresponding locations on all the other maps. The bottom layer of this diagram is the most important, for it represents the grid of a locational reference system (such as latitude and longitude) to which all the maps have been precisely registered. shown in **Figure 2.4**

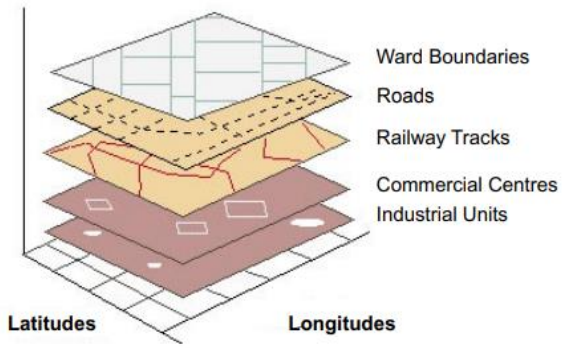


Figure 2.4. GIS an integrating technology

2.2.6 Navigation

Navigation is coordinated and goal – directed route following through space. It consists of two components: locomotion and way-finding. Locomotion is guidance through space in response to local sensorimotor information in immediate surrounds. It finds support surfaces, avoid obstacles and barriers, move through openings. Way-finding is planning and decision – making in response to non – local information, undertaken to reach goal.[6]

2.2.7 Universal Transverse Mercator (UTM)

UTM provides georeferencing at high levels of precision for the entire globe. Established in 1936 by the International Union of Geodesy and Geophysics, it is adopted by many national and international mapping agencies. It is commonly used in topographic and thematic mapping, for referencing satellite imagery and as a basis for widely distributed spatial databases. Universal Transverse Mercator (UTM) coordinates define two dimensional, horizontal, positions. Each UTM zone is identified by a number. UTM zone numbers designate individual 6° wide longitudinal strips extending from 80° South latitude to 84° North latitude as distortions at the poles is too large. Each zone has a central meridian. For example, Zone 14 has a central meridian of 99° west longitude. The zone extends from 96° to 102° west longitude. Locations within a zone are measured in meters eastward from the central meridian and northward from the equator. shown in **Figure 2.5**

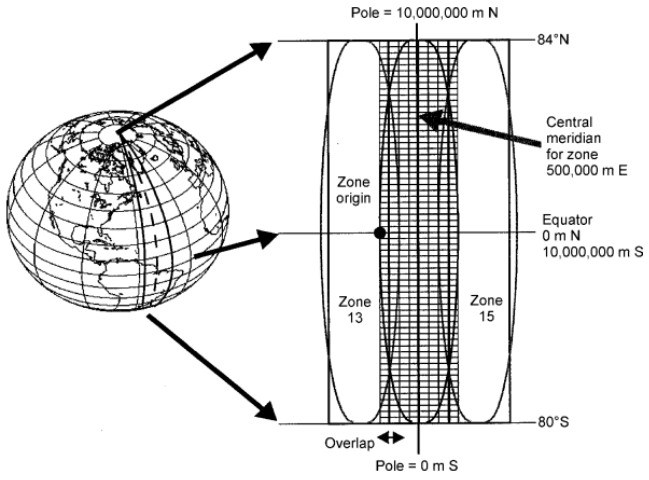


Figure 2.5. Overlap in UTM projection.

2.2.8. Latitude and Longitude

• Latitude

Two angles are sufficient to specify any location on the reference ellipsoid representing the Earth. Latitude is an angle between a plane and a line through a point. Imagine a flat plane intersecting an ellipsoidal model of the Earth. If the plane is coincident or parallel with the equator the result is a parallel of latitude. The equator is a unique parallel of latitude that also contains the center of the ellipsoid as shown in Figure 1.8. The equator is 0° latitude, and the North and South Poles are situated at +90° north and −90° south latitude, respectively. In other words, values for latitude range from a minimum of 0° to a maximum of 90°. shown in **Figure 2.7 [7]**

- **Longitude**

Longitude is an angle between two planes. It is a dihedral angle. In other words, it is an angle measured at the intersection of two planes that are perpendicular to the plane of the equator. In the case of longitude, the first plane passes through the point of interest, the place whose longitude you wish to know, and the second plane passes through an arbitrarily chosen point representing zero longitude. Today, that place is Greenwich, England. The measurement of angles of longitude is imagined to take place where the two planes meet, and that place is the line

known as the polar axis. As it happens, that line is also the axis of rotation of the aforementioned ellipsoidal model of the Earth. And where they intersect that ellipsoidal model they create an elliptical line on its surface. This elliptical line is then divided into two meridians at the polar axis. One half becomes a meridian of east longitude, which is labeled E or given a positive (+) values, and the other half a meridian of west longitude, which is labeled W or given a negative (−) value as shown in **Figure 2.6** **[7]**

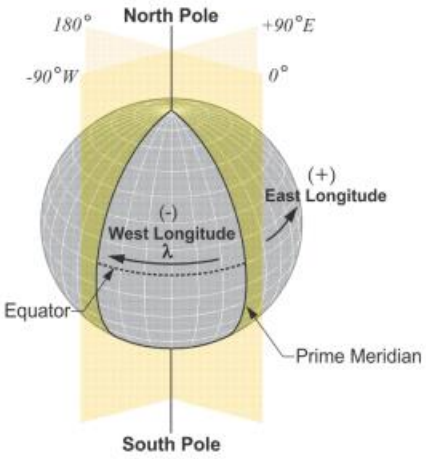


Figure 2.6. Longitude

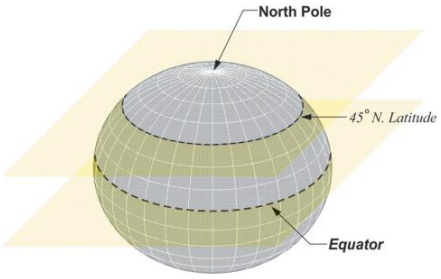


Figure 2.7. Latitude

2.3. Vehicle Coordinate System:

There are 6 degrees of freedom in the vehicle, as shown in the **Figure 2.8** This is fixed to the vehicle. X is longitudinal direction, Y is lateral, Z is vertical. Origin is at CG of the vehicle. The rotational motion along vehicle X is called roll, The rotational motion along vehicle Z is called Yaw, the rotational motion along vehicle Y is called pitch.

In this project we focus in three dimensions in motion first longitudinal, lateral motion and their equation and yaw stability motion.

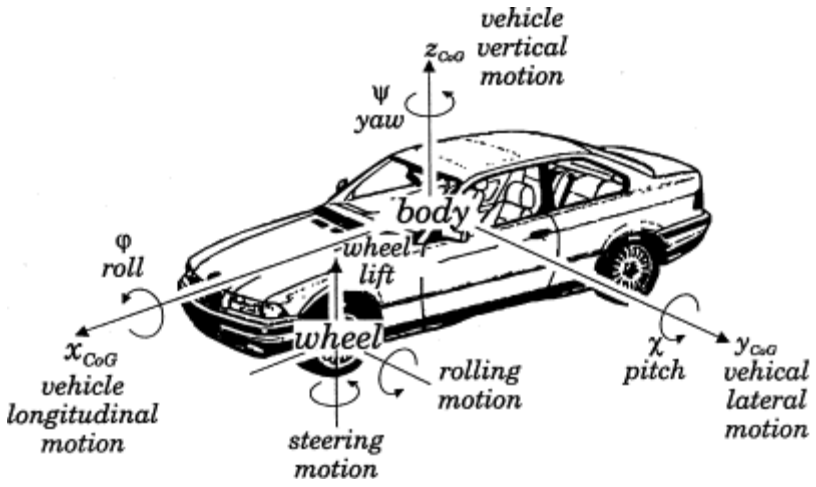


Figure 2.8. Vehicle Coordinate System 6 DOF

2.3.1. LONGITUDINAL VEHICLE DYNAMICS

Consider a vehicle moving on an inclined road as shown in **Figure 2.9**. The external longitudinal forces acting on the vehicle include aerodynamic drag forces, gravitational forces, longitudinal tire forces and rolling resistance forces.

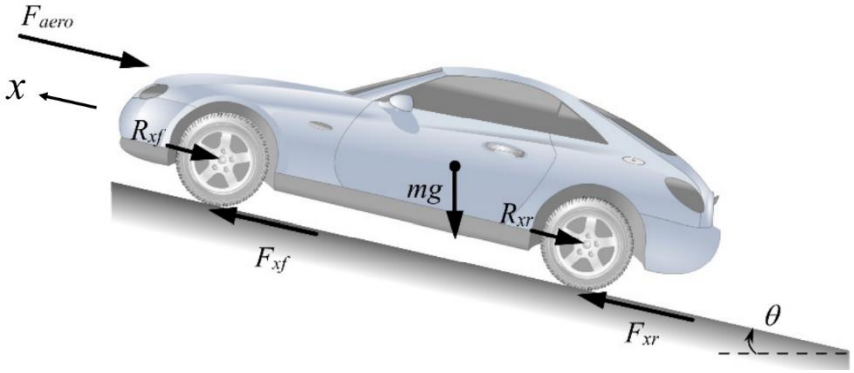


Figure 2.9 Longitudinal forces acting on a vehicle moving on an inclined road

Longitudinal forces acting on a vehicle moving on an inclined road A force balance along the vehicle longitudinal axis yields **[8]**

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg\sin(\theta)$$

where

F_{xf} is the longitudinal tire force at the front tires

F_{xr} is the longitudinal tire force at the rear tires

F_{aero} is the equivalent longitudinal aerodynamic drag force

R_{xf} is the force due to rolling resistance at the front tires

R_{xr} is the force due to rolling resistance at the rear tires

m is the mass of the vehicle

g is the acceleration due to gravity

θ is the angle of inclination of the road on which the vehicle is traveling

2.3.2. Lateral dynamics

For understand the lateral motion of the vehicle, it is critical to model the lateral dynamics. Lateral motion control is used in various ADAS features like lane centering, lane keeping etc. The lateral dynamics models are also imperative in modeling vehicle behavior's during the lateral maneuver's and can be used to study and design the system and components like we want to do in this project drive assistance system using GPS, GIS information and some data from vehicle.

2.3.3. Bicycle Model

A lot of handling vehicle dynamics models are available of various complexities and accuracy. One of the simplest and most commonly used models is the bicycle model. The term bicycle is because both the front wheels are taken as single entity and also both the rear wheels making it a two-wheel model.

2.3.4. Dynamic Bicycle Model of Lateral Vehicle

A bicycle model of the vehicle with two degrees of freedom is considered, as shown in **Figure 2.10.** The two degrees of freedom are represented by the vehicle lateral position y and the vehicle yaw angle ψ. The vehicle lateral position is measured along the lateral axis of the vehicle to the point O that is the center of rotation of the vehicle. The vehicle yaw angle ψ is measured with respect to the global X-axis. V denotes the longitudinal velocity of the vehicle at the Ignoring Road bank angle for now and applying Newton's second law for motion along the Y- axis.

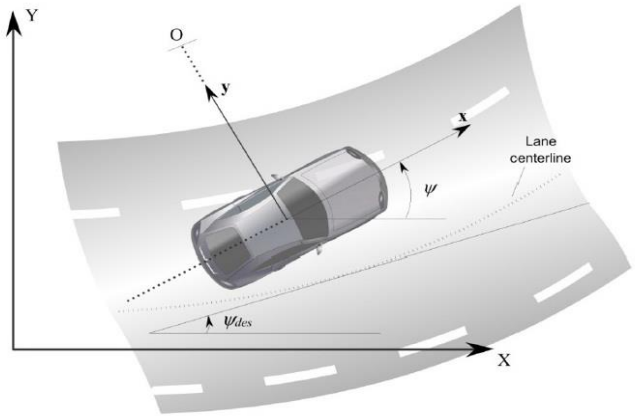


Figure 2.10. The lateral system in terms of rotating coordinates

$$ma_y = F_{yf} + F_{yr}$$

After some mathematical diversion we have a mathematical modal ready to use it.**[8]**

$$\frac{d}{dt}\begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{2C_{\alpha f} + 2C_{\alpha r}}{mV_x} & 0 & -V_x - \dfrac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{2\ell_f C_{\alpha f} - 2\ell_r C_{\alpha r}}{I_z V_x} & 0 & -\dfrac{2\ell_f{}^2 C_{\alpha f} + 2\ell_r{}^2 C_{\alpha r}}{I_z V_x} \end{bmatrix} \begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix}$$

$$+ \begin{Bmatrix} 0 \\ \dfrac{2C_{\alpha f}}{m} \\ 0 \\ \dfrac{2\ell_f C_{\alpha f}}{I_z} \end{Bmatrix} \delta$$

NOMENCLATURE

F_y	lateral tire force
F_{yf}	lateral tire force on front tires
F_{yr}	lateral tire force on rear tires
V_x	longitudinal velocity at c.g. of vehicle
V	total velocity at c.g. of vehicle
$\dot{y}$	lateral velocity at c.g. of vehicle
V_y	lateral velocity at c.g. of vehicle (same as $\dot{y}$)
m	total mass of vehicle
I_z	yaw moment of inertia of vehicle
ℓ_f	longitudinal distance from c.g. to front tires
ℓ_r	longitudinal distance from c.g. to rear tires
L	total wheel base ($\ell_f + \ell_r$)
ψ	yaw angle of vehicle in global axes
$\dot{\psi}$	yaw rate of vehicle
r	yaw rate of vehicle (same as $\dot{\psi}$)
X , Y	global axes
δ	steering wheel angle
δ_f	front wheel steering angle
δ_r	rear wheel steering angle
δ_o	steering angle of outer wheels
δ_i	steering angle of inner wheels
ℓ_w	track width
α_f	slip angle at front tires
α_r	slip angle at rear tires
C_α	cornering stiffness of tire

2.4. Engine Control Unit

2.4.1 Introduction

Modern automobiles consist of a number of different computer components, called Electronic Control Units (ECUs). Each automobile contains from 20-100 of these devices, with each ECU being responsible for one or more particular features of the vehicle **[9]**

In order to make the vehicles smart and more safety new ECUs or nodes were added to the vehicles. It is not a problem. The problem was how to connect between these different ECUs to deliver the data from ECUs to another.

2.4.2. Controller Area Network

The Controller Area Network, or the CAN bus, is a network used in many vehicles today as it was standardized. It has been around a long time, and handles the internal communications between electronic control units.

Each ECU can communicate with all other ECUs using the CAN bus system, which eliminates the need for complex dedicated wiring. Specifically, an ECU can use the CAN bus to prepare and broadcast information (such as sensor data) (consisting of two wires, CAN low and CAN high) **Figure 2.11**. All other ECUs on the CAN network accept the broadcasted data, and each ECU can then check the data and decide whether to receive or ignore it.**[10]**

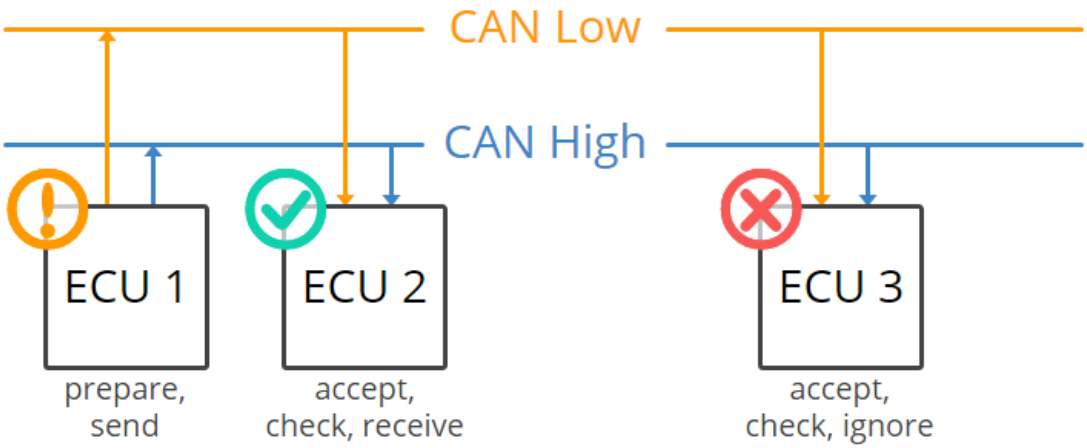


Figure 2.11. CAN Bus

2.4.3 How do CAN bus modules communicate?

For communication, the CAN bus employs two separate cables. CAN high and CAN low are the names of the cables.

Both lines carry 2.5V when the CAN bus is in idle mode. When data bits are sent, the CAN high line rises to 3.75V and the CAN low line falls to 1.25V, resulting in a 2.5V difference between the lines **Figure 2.12**. The CAN bus is not vulnerable to inductive spikes, electrical fields, or other noise because communication is based on a voltage difference between the two bus lines. As a result, the CAN bus is a solid option for networked communications on mobile equipment.**[11]**

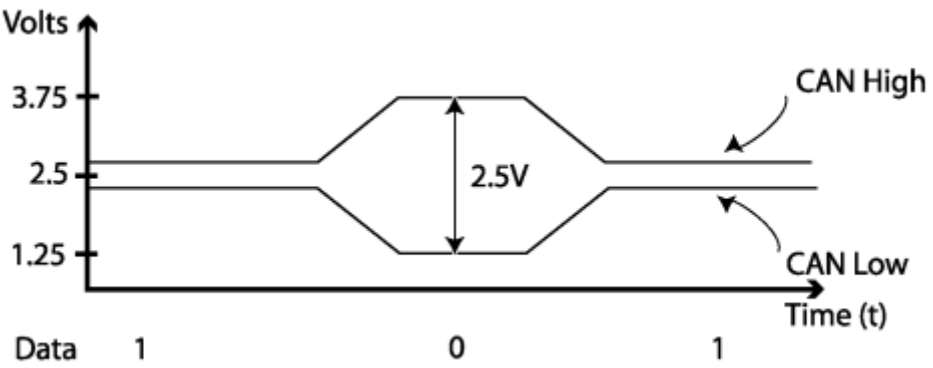


Figure 2.12. Differential Between CAN High and CAN Low

2.4.4 Autonomous Vehicle

An Autonomous Vehicle (AV) is a vehicle that can guide itself, as opposed to being controlled by human. The AV is a kind of driverless vehicle that has become in reality and is the art of driving using computers for future. AVs have been targeted due to: **1.** increasing vehicle safety, **2.** reduction of accidents, **3.** reduction of fuel consumption, **4.** releasing of driver time and business opportunities, **5.** new potential market opportunities, and **6.** reduced emissions and dust particles. **[12]**

In general, Autonomous Vehicle (AV) needs autonomous mobile navigation to find its:
1. localization
2. map building,
3. path planning, and
4. path tracking. In addition, it is required the AV obstacle avoidance through detection and classification.

2.5. Literature Review

1. GPS tracking system for autonomous vehicles

The propose of this research paper is to be able to memorize a route based on Global Positioning System (GPS) by using a mechatronics system, rather than using pre-saved maps that are infrequently updated and do not include all roads of all countries. Experimental tests are conducted using a small-scale car equipped with the proposed mechatronics system.

In order to navigate a certain path, the driver has to drive the vehicle on the desired path only, and by using GPS to determine the current location, we can determine the distance to the next waypoint **Figure 2.13**, and so we can calculate the suitable speed and steering angle before the next waypoint.**[13]**

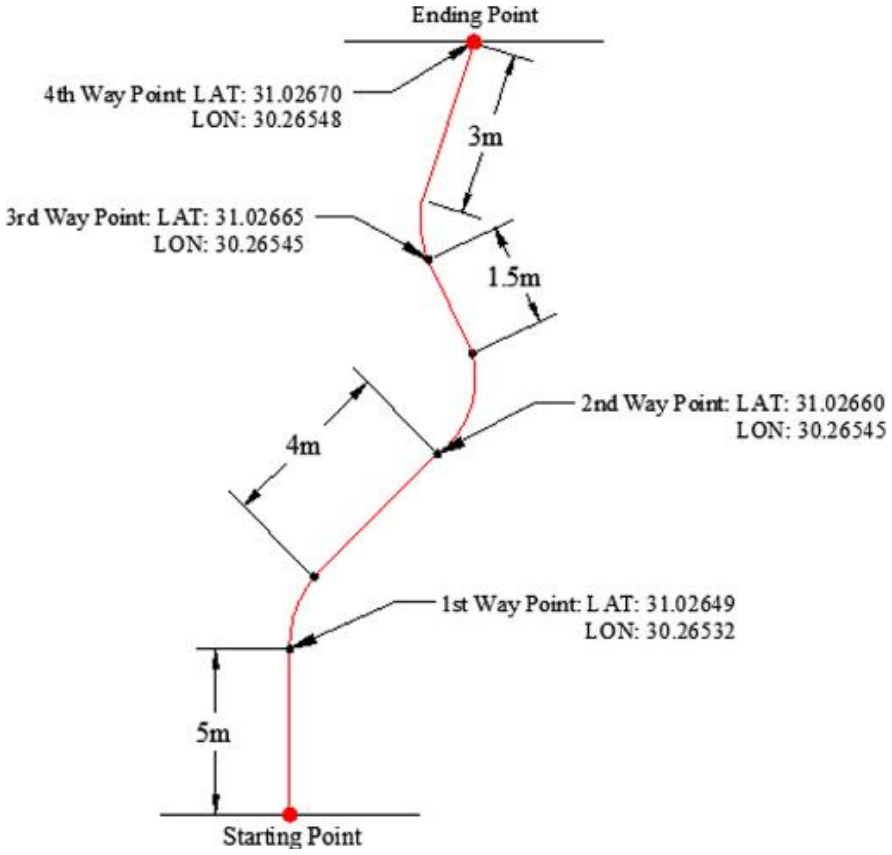


Figure 2.13. a small full path example

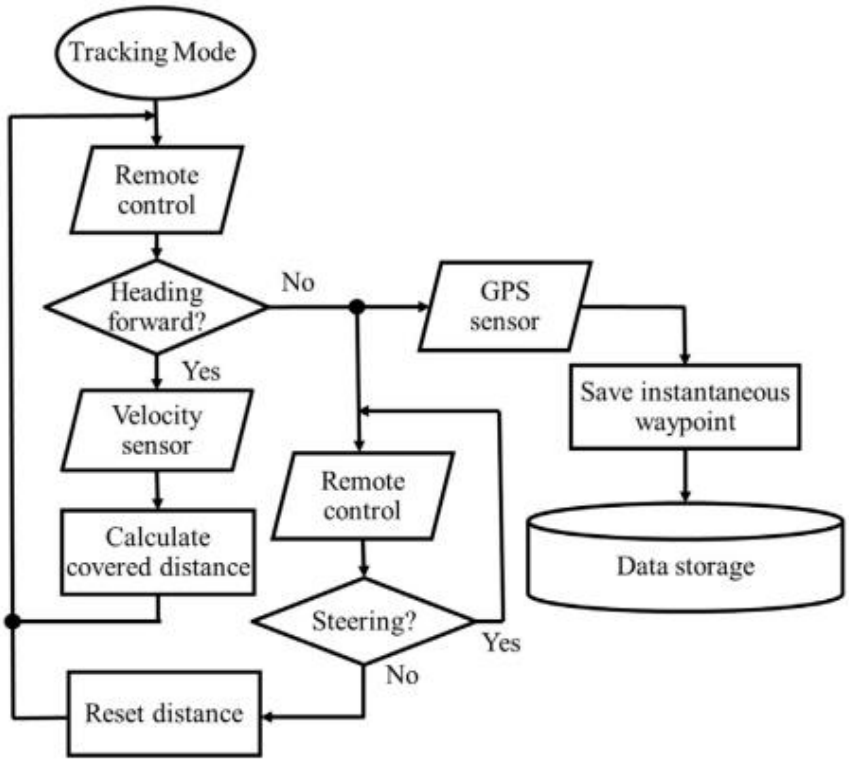


Figure 2.14. Flowchart of control sequence of the tracking mode

2. Developing Training Module on GPS Applications in Automotive Safety and Stability 2016

In this project contents allow the trainee to reach the know-how of this technology in addition to the future trends related to automotive technologies. The content includes acquiring GPS signals, using MATLAB routine to transfer data to GIS for determining the track, identifying the driving track curvature segments and estimating the safe driving speed using simplified vehicle stability model, and finally alarming the driver.

This project will address this engineering issue by considering vehicle stability system and navigation system on roads, developing a system consist of GPS sensor, microcontroller (Arduino) and a processor which will be a personal laptop by using MATLAB as a programming tool for achieving the prototype aims. This, of course, will help in decreasing accidents and save people lives when the target control system is adopted and applied in vehicles as a final product system, and it will help in autonomous cars researches**. [14]**

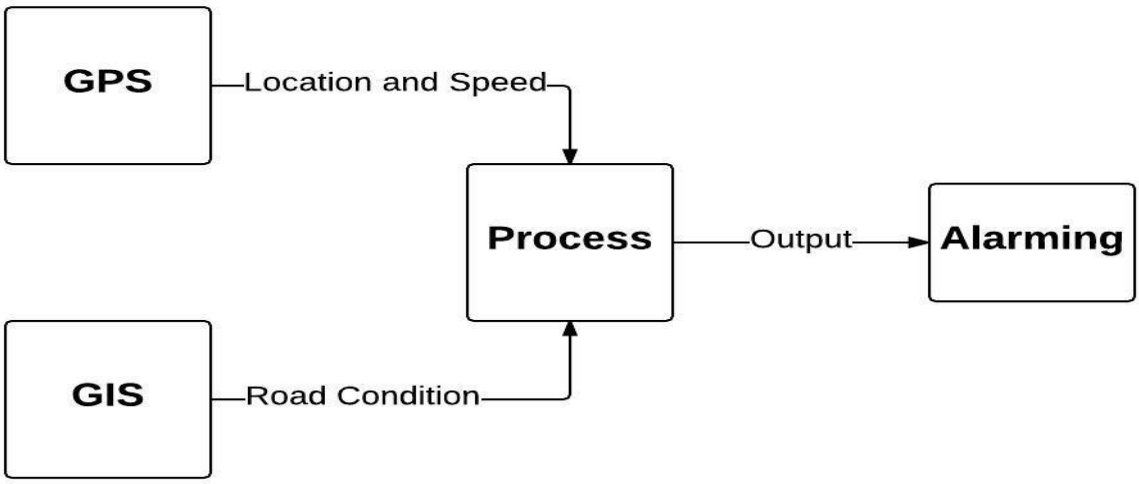


Figure2.15. Basic concepts of safety road speed distance

3

CHAPTER 3

3. System Design

3.1 Introduction

This chapter introduces the design part of project. It provides a description to the different hardware options, the different components of the system, the logic flow and the interconnection of the components.

3.1. 2 System Block Diagram

In general, any system contains three basic concepts (input, processing, and output).
Input – anything you do to activate the system or give the system to use. And Process the actual steps and function the system will perform. Then comes the stage. Output the result after doing all the steps. **Figure 3.1** shows these concepts of our project in a simple block diagram.

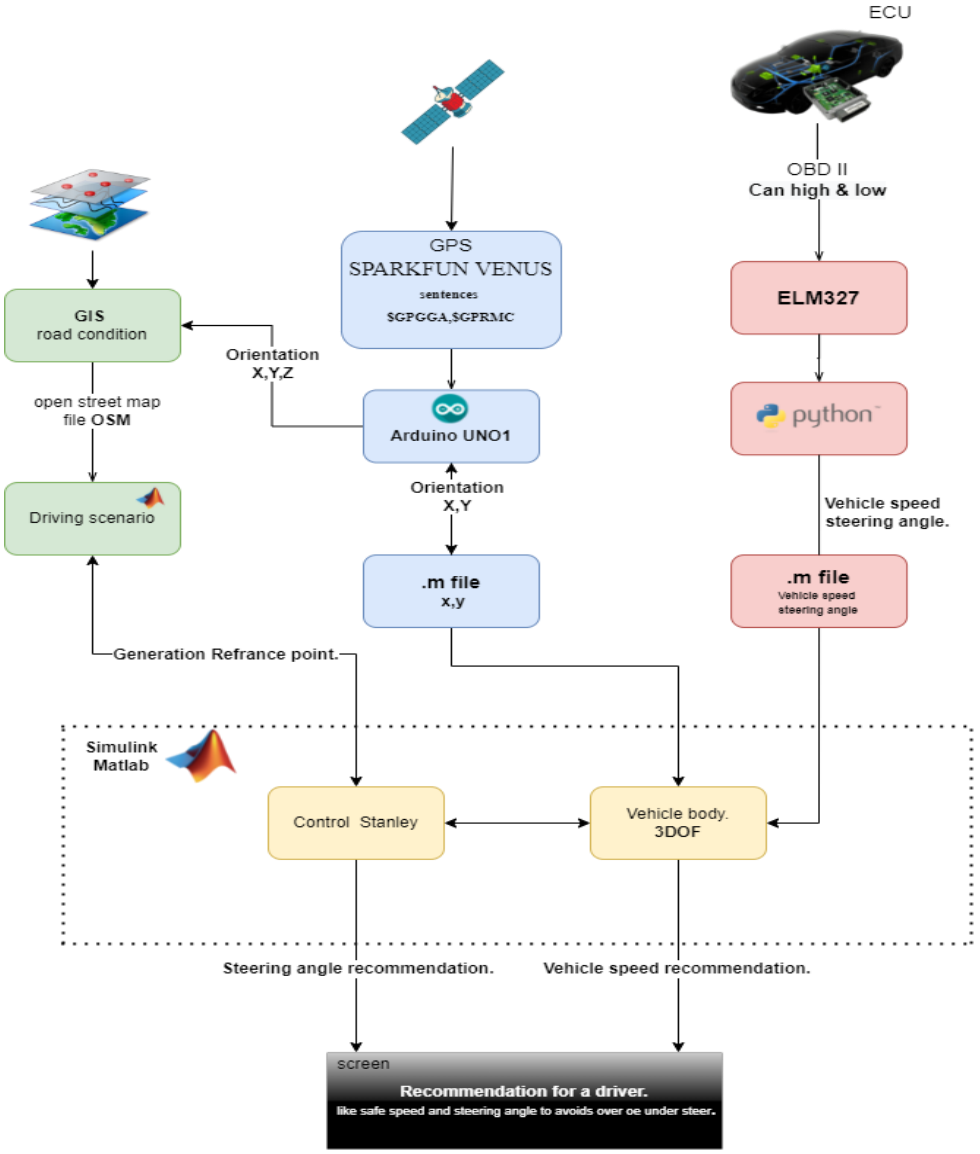


Figure 3.1 .the simple block diagram for our project procedure

3.2 Input

In this project, the system will receive data from three subsystems that are all working at the same time: GPS sensor, GIS data, and ECU data.

3.2.1 GPS sensor

We will use it to receive the vehicle velocity and position in NMEA format (National Marine Electronics Association.) with respect to WGS 84(World Geodetic System) and with respect to GIS data which we will talk about later. And we will convert the NMEA format into UTM format (x, y, z).

To get good accuracy of the location determination we need to take in mind the update rate of the GPS sensor and its number of channels.

- **Update rate**

Update rate is the number of times per second that will receive current position. A higher update rate decreases lag time and improves distance measurements and tracking especially when moving on a curvy route. For low-speed applications, an update rate of 0.1Hz is sufficient whereas 5 or even 10 Hz update rate is required for other high-speed navigation. As the speed is increased, more update rate is required to reduce blind area between two updates. **Table 3.1** below shows blind distances for a speed of 100km/h. with different update rate.

Table 3.1 Distance moved before next update (Speed = 100km/h = 27.778m/s)

Update rate (Hz)	Distance (cm)
1	2777.778
5	555.556
10	277.778
20	138.889

- **Number of channels**

A GPS receiver is usually described by its number of channels. A receiver may have 6 channels, 12 channels, or hundreds of channels. Each channel trying to communicate with only one satellite, as the number of channels increases the error will decrease. In this project the GPS sensor that will be used is "**SparkFun Venus GPS"** with update rate 1 Hz, which will be controlled by Arduino Uno microcontroller, the sensor data sheet will be attached in Appendix A of this book.

- **GPS test**

Practical experiment to make sure that the sensors work and the speed of reading takes place in accordance with changing circumstances. In the first experiment, the path inside the university has been taken, walking along a random path along a random length. We noticed that seven readings are taken of the same coordinates per second. **Figure 3.2** shows first experiment

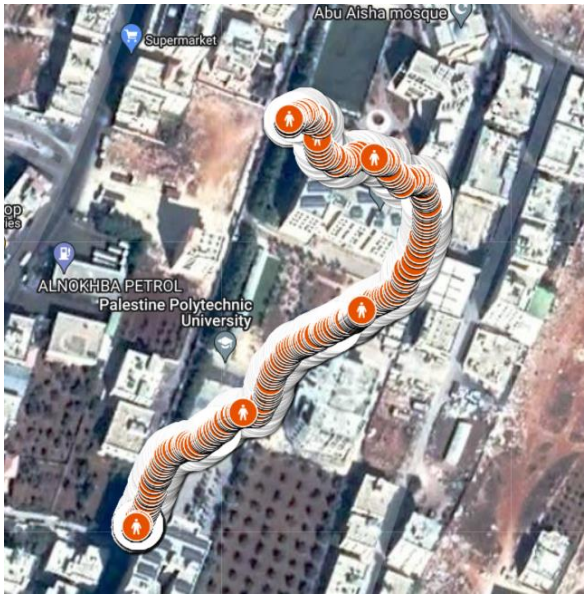


Figure 3.2. first experiment

In the second experiment, from the Wad-alqaf Reserve, Safa Halhoul Street, and take the starting point and the end point in this path, taking into account that the area does not have confusion such as high buildings and some influencing factors. As we talked about earlier, I fixed the speed at 50 kilometres per hour we noticed that the number of readings per second differed in the speed difference, although the speed was fixed. The number of readings was taken differently. Seven readings were taken bearing the same coordinates. Another time was taken six times to four times. The probable cause was the presence of noise due to the speed of moving from one location to another. **Figure 3.3** shows second experiment

Figure 3.3 shows second experiment

- **Hardware recommendation.**

Spark Fun Venus GPS module is shown in the **figure 3.4.**

- This module has an external antenna and built-in EEPROM.
- Interface: RS232 TTL
- Power supply: 3.3v
- Default baud rate: 9600 bps
- Works with standard NMEA sentences

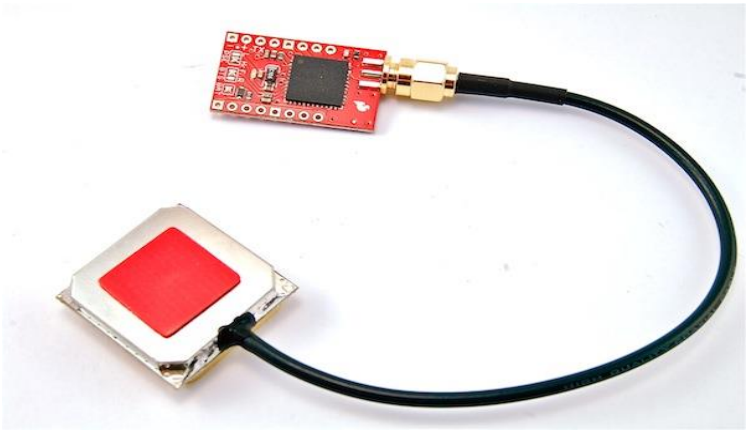


Figure 3.4. Spark Fun Venus GPS

Table 3.2 Connect GPS sensor to Arduino

Spark fun GPS Module	**Wiring to Arduino UNO**
VCC	3.3V
TX	TX pin defined in the software serial "Digital pin(11)"
RX	RX pin defined in the software serial "Digital pin(10)"
GND	GND

In this project the sentences that will be used are the sentences stars with these prefixes $GPGGA,$GPRMC, $PGRMZ by using these sentences we can import longitude, latitude, altitude.

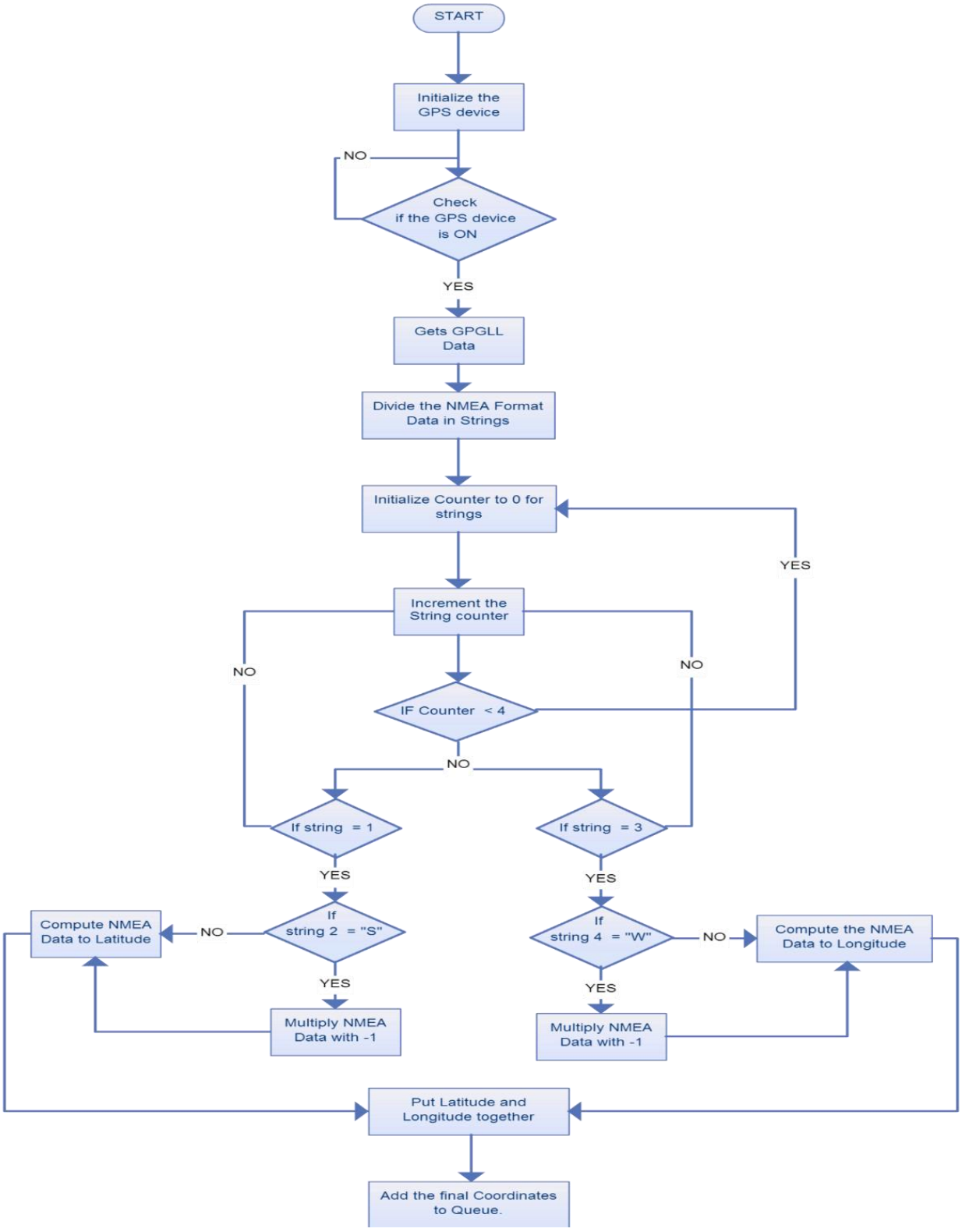


Figure 3.5. GPS Navigation Systems Flowchart

3.2.2 GIS Data

In this project, GIS data is used to translate and process street maps and take the coordinates and information to be obtained, and the street point's coordinates import in a Universal Transverse Mercator format (UTM) which is a coordinate system uses a 2-dimensional cartesian coordinate system to give locations in meters on the surface of the earth with relative to the equator and greenwich as an origin.

In the beginning, determined the requirements that needed for the path to be taken, where it has been focused on that the required road contains bends and slopes, then decided to take the Safa - Halhul road, after that determined the starting and ending points of the road, in order to draw the path on the AutoCAD program and determine the value of the radius and the length of the path, as well as specifying the coordinates of each point separately in the path.

In summary, the current position of the vehicle coming from the GPS. it pair with the maps to take important information such as cornering and the radius of the street corner. Then send it to the model to produce the required recommendations. Figure 3.6 shows the information needed from the path, such as the radius of the bend and the coordinates between each point.

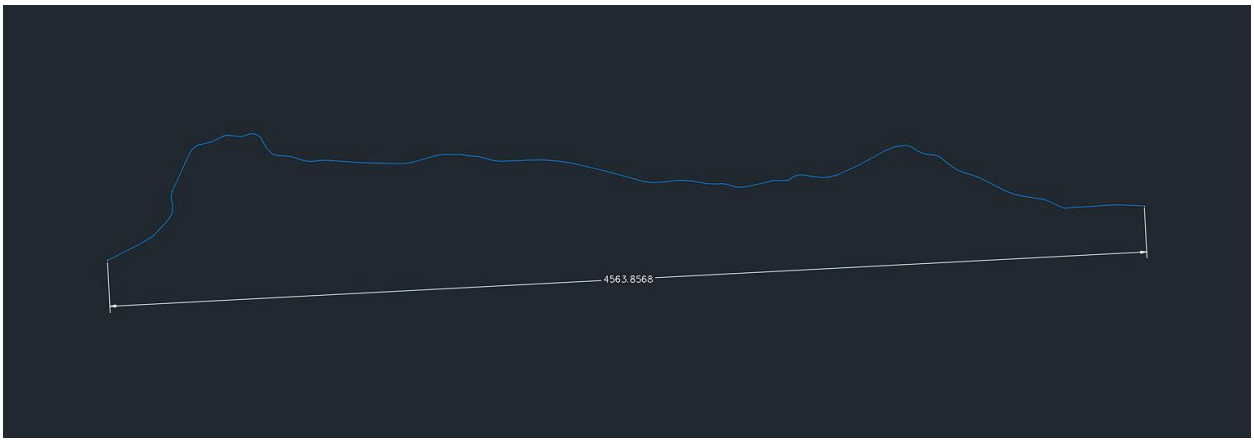


Figure 3.6. Drawing Track By using auto cad to take dimensions

3.2.3 ECU Information

In this section we will discuss how to collect real-time information from a vehicle control unit, we can collect information via OBD-II, which has become standard in most of all cars. OBD-II protocols over a CAN bus that will allow the real-time variables visualization for the vehicle state.

Measuring Modes

The external device requests are divided into 9 services as shown **in table 3.4** In order to request data, it is necessary to use PID's, each PID corresponding to different vehicle data information, The PID samples that will be used in our project are shown in **table 3.5**

Table 3.3. The PID samples

Diagnostic service number	Diagnostic service
01	Request current powertrain diagnostic data
02	Request current powertrain diagnostic data
03	Request emission-related diagnostic trouble codes
04	Clear/reset emission-related diagnostic information
05	Request oxygen sensor monitoring test results
06	Request on-board monitoring test results for specific monitored systems
07	Request emission-related diagnostic trouble codes detected during current or last completed driving cycle
08	Request control of on-board system, test or component
09	Request vehicle information

Table 3.4 PID'S OF Vehicle speed and steering angle

parameter	**service and PID (hex)**
Vehicle speed	01 0D
RPM	01 0C

- **CAN Bus Shield for Arduino**

Due to its relatively broad reach, communication speed, and high dependability, CAN is one of the most widely used bus communication protocols. It's widely used in control machines and diagnostics buses for automobiles. The CAN bus shield board uses the CAN MCP2515 controller with SPI interface to give CAN communication to the Arduino. This one is in charge of controlling the CAN message pre-processing, as well as having a CAN MCP2551 transceiver to handle the bus electric interface **Figure 3.6.**

Features

• It implements CAN 2.0B at speeds up to 1 Mbps(megabits per second);
• It has a SPI interface capable to operate at 10 MHz;
• It supports standard (11 bits), extended (29 bits), and remote frames;
• It has two receiving buffers for storing priority messages**,** a 9-pin sub-D industrial standard, and 2 LED indicators.
• Its operating voltage is 5 V, its dimensions are 68 x 53 mm, and its weight is only 50g.

Figure 3.6. CAN Bus Shield

- **ELM 327**

There are numerous types of interfaces available for the ELM327 OBD, however the microcontroller chip developed by ELM Electronics is the most widely used. The ELM327, which supports all OBD protocols including KWP, PWM, ISO, and CAN, is the most extensively used in actual use. Its design is minimal power. Depending on the manufacturer and the terminal type, it can be purchased for as little as $3 to $20.

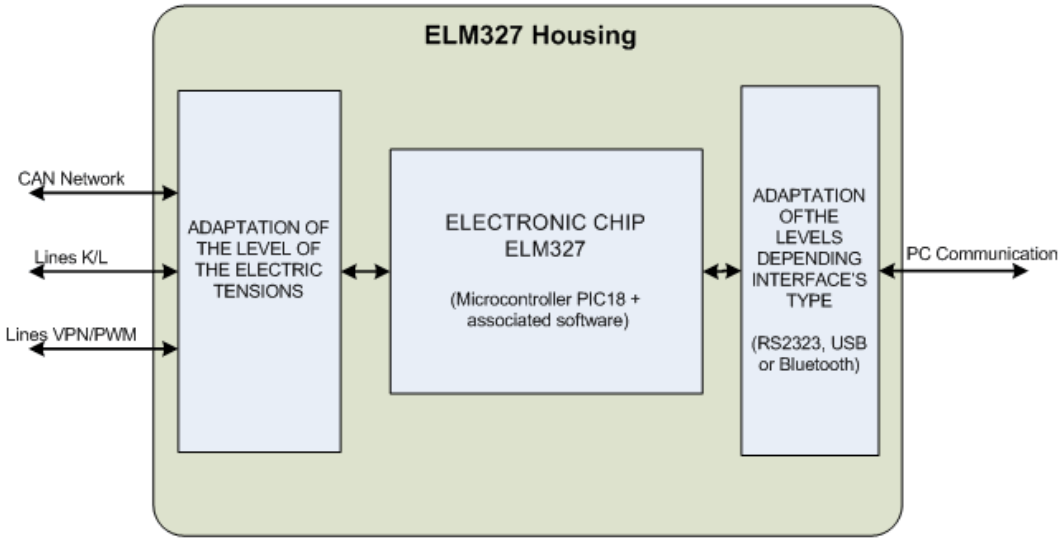


Figure 3.7 . The diagram below shows the main function of an interface.

- ✓ **Type of ELM327 interfaces**

1. **ELM327 RS232 (RS or Series):** Modern PCs are rapidly phasing out this sort of output. The price is lowest.
2. **ELM327 USB:** Needs a USB driver to be installed; the tethered connection is hardwired as opposed to wireless; advantages: All PCs come with a USB port. Given that it might be faster and more secure.
3. **ELM327 Bluetooth:** It has the benefit of being wireless and may be used with technology such as a computer or a smartphone like Wi-Fi; however, battery consumption should be taken into consideration.
4. **ELM327 Wi-Fi:** Because it has a wireless connection, any technology, like a computer or a smartphone, can utilize it. Additionally, the Wi-Fi interface's rapid, simple, and other benefits when using an iPhone or iPad

The decision to utilize a Bluetooth adaptor was made since it is less expensive and often uses less power than a Wi-Fi adaptor.

Figure 3.8 . Type of ELM327 interfaces

Table 3.5 Comparison between ELM327 and Seeed CAN Bus shield

features	SEEED CANBUS SHEILD V2	SEED CANBUS SHEILD V1.2	SUNFLOWER CAN SHEILD	ELM 327
microprocess	MCP 2515	MCP 2521	MCP 2521	ELM 327
Compatibility with Arduino boards	Designed specifically for use with Arduino boards	Designed specifically for use with Arduino boards	Designed specifically for use with Arduino boards	Can be used with Arduino boards via a USB to Serial adapter or with a Bluetooth or WIFI module
Coding language.	C	C	C	python
operating voltage	5V	5V	5V	4.5V to 5.5V
communication interface	USB	USB	USB	USB, Bluetooth, WiFi
Compatibility with other platforms	Not designed for use with other platforms	Not designed for use with other platforms	Not designed for use with other platforms	Can be used with other platforms through

				a USB to Serial adapter or with a Bluetooth or WiFi module
supported protocol	SAE J1850 PWM (41.6 kbit/s)/ SAE J1850 VPW (10.4 kbit/s)/ ISO 9141-2 (5 baud init, 10.4 kbit/s) / ISO 14230-4 KWP (5 baud init,10.4 kbit/s)/ ISO 14230-4 KWP (fast init, 10.4 kbit/s)/ ISO 15765-4 CAN (11 bit ID, 500 kbit/s)/	SAE J1850 PWM (41.6 kbit/s)/ SAE J1850 VPW (10.4 kbit/s)/ ISO 9141-2 (5 baud init, 10.4 kbit/s) / ISO 14230-4 KWP (5 baud init,10.4 kbit/s)/ ISO 14230-4 KWP (fast init, 10.4 kbit/s)/ ISO 15765-4 CAN (11 bit ID, 500 kbit/s)/	SAE J1850 PWM (41.6 kbit/s)/ SAE J1850 VPW (10.4 kbit/s)/ ISO 9141-2 (5 baud init, 10.4 kbit/s) / ISO 14230-4 KWP (5 baud init,10.4 kbit/s)/ ISO 14230-4 KWP (fast init, 10.4 kbit/s)/ ISO 15765-4 CAN (11 bit ID, 500 kbit/s)/	SAE J1850 PWM (41.6 kbit/s)/ SAE J1850 VPW (10.4 kbit/s)/ ISO 9141-2 (5 baud init, 10.4 kbit/s) / ISO 14230-4 KWP (5 baud init,10.4 kbit/s)/ ISO 14230-4 KWP (fast init, 10.4 kbit/s)/ ISO 15765-4 CAN (11 bit ID, 500 kbit/s)/
Cost	34$	34$	34$	20$

3.3. Processes

In this part it will be explained the processes of project as shown in block diagram 3.1 after getting the data from GPS receiver to Arduino Uno board, GIS data about rod specifications and have the required data from ECU of vehicle, then making the process for data and have the output to display to driver. **Figure 3.7** Block diagram for process

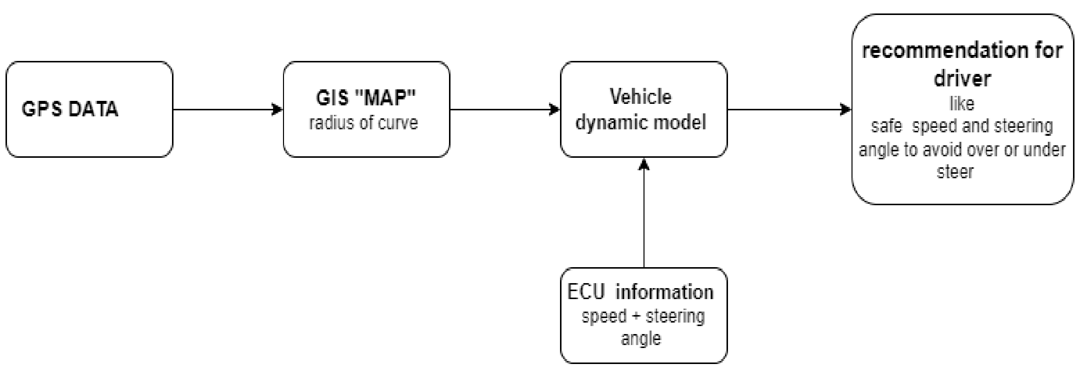


Figure 3.7. Block diagram for process

For more accurate results and better processing, a PC will be used to overcome the electrical problems, for achieving our project aims easier and faster.

In addition, the laptop provides an excellent programing tool which is MATLAB program; this program provides us flexibility, accessibility, and the ability for doing any coding algorithms with very simple language.

3.4. Outputs

The output of this system will be warning massage asking the driver to decrease the vehicle speed under the safe speed before reaching the corner and adjustment to steering angle increased or decreased the steer value this massage will appear before the corner with an appropriate time, in addition there will be sound warning to keep the driver attention with the track. **Figure 3.8** shows Flowchart for warning massage.

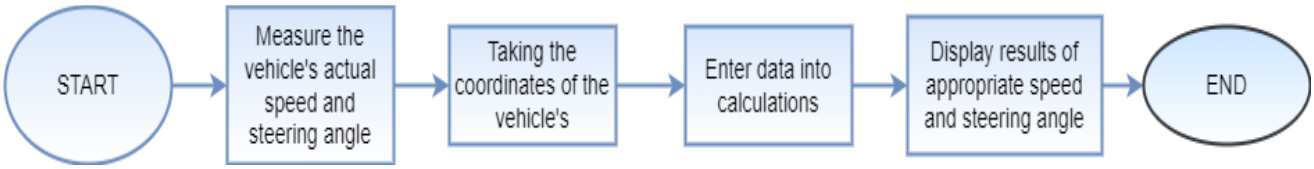


Figure 3.8. Flowchart for warning massage.

3.5 Table of cost

4

Chapter 4

Simulation

Introduction:

The first step to apply the project in real ward is apply it in digital ward, in this project the simulation it will be done by MATLAB program by using coding, Simulink and driving scenario application in MATLAB to achieve the goal.

By the Simulink have the main Blok that receive the input data to make the calculation and the driving scenario prepare the GIS file and the specific data for the road and save it to use it in MATLAB, by coding it can request the file of road data and other file like reference throttle position.

4.1. Simulink:

4.1.1 Stanley Controller Block:

We're going to look at the Stanley Controller itself. We'll take a look at its implementation, starti g with generation of waypoints, and then moving on to actually implementing this, and building the models in Matlab, and of course doing all of these visualizations of the vehicle motion. For those of you that don't know, the Stanley Controller has been around for a little while, and it's actually a path tracking algorithm. It was actually first used in real life in the Stanford racing team in the DARPA GRAND Challenge, and it's using and computing the steering wheel angle to follow a reference trajectory. **Figure 4.1**

The path tracking approach is used by Sanford University's Darpan Grand Challenge team. Different from the pure pursuit method using the rear axle as its reference point, Stanley method use the front axle as its reference point. Meanwhile, it looks at both the heading error and cross-track error. In this method, the cross-track error is defined as the distance between the closest point on the path with the front axle of the vehicle. [12]

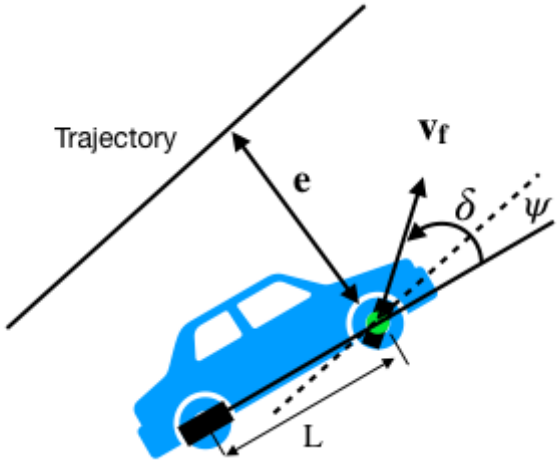


Figure 4.1. Geometric path tracking

In this method, the center of the rear axle is used as the reference point on the vehicle.

4.1.2 Vehicle Body 3DOF Dual Track

The Vehicle Body 3DOF block implements a rigid two-axle vehicle body model to calculate longitudinal, lateral, and yaw motion. The block accounts for body mass and aerodynamic drag between the axles due to acceleration and steering. As shown in **Figure 4.3**. To determine the vehicle motion, the block implements these equations for the dual track.

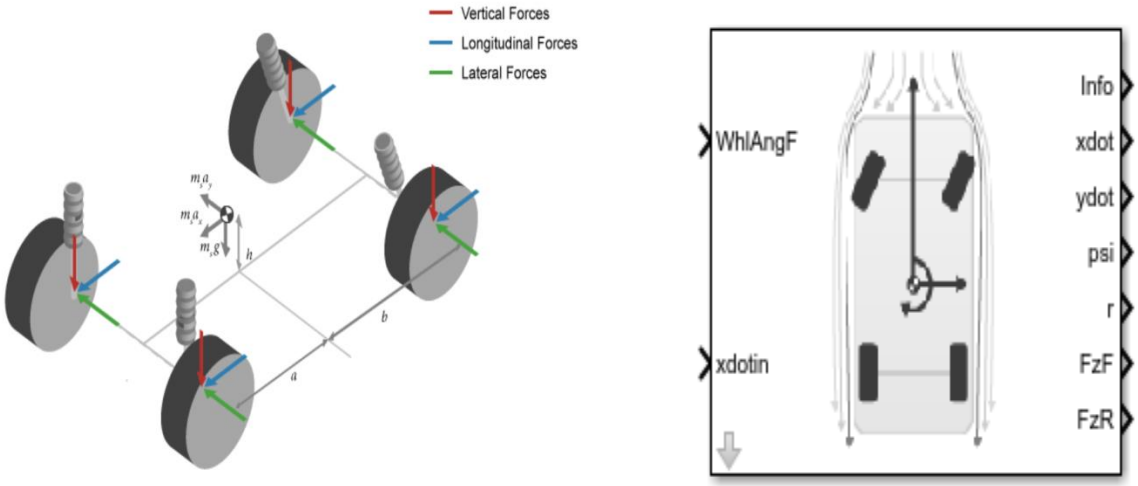


Figure 4.3. The Vehicle Body 3DOF block **[13]**

The block uses these equations to calculate the rigid body planar dynamics:

$$\ddot{x} = \dot{y}r + \frac{F_{xfl} + F_{xfr} + F_{xrl} + F_{xrr} + F_{xext}}{m}$$

$$\ddot{y} = -\dot{x}r + \frac{F_{yfl} + F_{yfr} + F_{yrl} + F_{yrr} + F_{yext}}{m}$$

$$\dot{r} = \frac{a(F_{yfl} + F_{yfr}) - b(F_{yrl} + F_{yrr}) + \frac{w_f(F_{xfl} - F_{xfr})}{2} + \frac{w_r(F_{xrl} - F_{xrr})}{2} + M_{zext}}{I_{zz}}$$

$$r = \dot{\psi}$$

If we set **Axle forces** to External longitudinal velocity, the block assumes a quasi-steady state for the longitudinal acceleration.

$$\ddot{x} = 0$$

For External forces

External forces include both drag and external force inputs. The forces act on the vehicle CG:

$$F_{x,y,z\ \ ext} = F_{d\ \ x,y,z} + F_{x,y,z\ \ input}$$

$$M_{x,y,z\ \ ext} = M_{d\ \ x,y,z} + M_{x,y,z\ \ input}$$

If we set Axle forces to External longitudinal forces, the block uses these equations:

$$F_{xflt} = F_{xflinput}$$

$$F_{yflt} = -C_{yfl}\alpha_{fl}\mu_{fl}\frac{F_{zfl}}{2F_{znom}}$$

$$F_{xfrt} = F_{xlrinput}$$

$$F_{yfrt} = -C_{yfr}\alpha_{fr}\mu_{fr}\frac{F_{zfr}}{2F_{znom}}$$

$$F_{xrlt} = F_{xrlinput}$$

$$F_{yrlt} = -C_{yrl}\alpha_{rl}\mu_{rl}\frac{F_{zrl}}{2F_{znom}}$$

$$F_{xrrt} = F_{xrrinput}$$

$$F_{yrrt} = -C_{yrr}\alpha_{rr}\mu_{rr}\frac{F_{zrr}}{2F_{znom}}$$

For Tire forces

The block uses the ratio of the local and longitudinal and lateral velocities to determine the slip angles:

$$\alpha_{fl} = atan\left(\frac{\dot{y} + ar}{\dot{x} + r\frac{w_f}{2}}\right) - \delta_{fl}$$

$$\alpha_{fr} = atan\left(\frac{\dot{y} + ar}{\dot{x} - r\frac{w_f}{2}}\right) - \delta_{fr}$$

$$\alpha_{rl} = atan\left(\frac{\dot{y} - ar}{\dot{x} + r\frac{w_r}{2}}\right) - \delta_{rl}$$

$$\alpha_{rr} = atan\left(\frac{\dot{y} - ar}{\dot{x} - r\frac{w_r}{2}}\right) - \delta_{rr}$$

The block uses the steering angles to transform the tire forces to the vehicle-fixed frame:

$$F_{xf} = F_{xft}\cos(\delta_f) - F_{yft}\sin(\delta_f)$$
$$F_{yf} = -F_{xft}\sin(\delta_f) + F_{yft}\cos(\delta_f)$$
$$F_{xr} = F_{xrt}\cos(\delta_r) - F_{yrt}\sin(\delta_r)$$
$$F_{yr} = -F_{xrt}\sin(\delta_r) + F_{yrt}\cos(\delta_r)$$

Table 4.1 The equations use these variables (Stanley Controller and Vehicle Body 3DOF)

$x, \dot{x}, \ddot{x}$	Vehicle CG displacement, velocity, and acceleration, along the vehicle-fixed *x*-axis
$y, \dot{y}, \ddot{y}$	Vehicle CG displacement, velocity, and acceleration, along the vehicle-fixed *y*-axis
Ψ	Rotation of the vehicle-fixed frame about the earth-fixed *Z*-axis (yaw)
$r, \dot{\Psi}$	Vehicle angular velocity, about the vehicle-fixed *z*-axis (yaw rate)
F_{xf}, F_{xr}	Longitudinal forces applied to front and rear wheels, along the vehicle-fixed *x*-axis
F_{yf}, F_{yr}	Lateral forces applied to front and rear wheels, along vehicle-fixed *y*-axis
$F_{xext}, F_{yext}, F_{zext}$	External forces applied to vehicle CG, along the vehicle-fixed *x*-, *y*-, and *z*-axes
F_{dx}, F_{dy}, F_{dz}	Drag forces applied to vehicle CG, along the vehicle-fixed *x*-, *y*-, and *z*-axes
$F_{xinput}, F_{yinput}, F_{zinput}$	Input forces applied to vehicle CG, along the vehicle-fixed *x*-, *y*-, and *z*-axes
$M_{xext}, M_{yext}, M_{zext}$	External moment about vehicle CG, about the vehicle-fixed *x*-, *y*-, and *z*-axes
M_{dx}, M_{dy}, M_{dz}	Drag moment about vehicle CG, about the vehicle-fixed *x*-, *y*-, and *z*-axes
$M_{xinput}, M_{yinput}, M_{zinput}$	Input moment about vehicle CG, about the vehicle-fixed *x*-, *y*-, and *z*-axes
I_{zz}	Vehicle body moment of inertia about the vehicle-fixed *z*-axis
F_{xft}, F_{xrt}	Longitudinal tire force applied to front and rear wheels, along the vehicle-fixed *x*-axis
F_{yft}, F_{yft}	Lateral tire force applied to front and rear wheels, along vehicle-fixed *y*-axis
F_{xfl}, F_{xfr}	Longitudinal force applied to front left and front right wheels, along the vehicle-fixed *x*-axis

F_{yfl}, F_{yfr}	Lateral force applied to front left and front right wheels, along the vehicle-fixed *y*-axis
F_{xrl}, F_{xrr}	Longitudinal force applied to rear left and rear right wheels, along the vehicle-fixed *x*-axis
F_{yrl}, F_{yrr}	Lateral force applied to rear left and rear right wheels, along the vehicle-fixed *y*-axis
F_{xflt}, F_{xfrt}	Longitudinal tire force applied to front left and front right wheels, along the vehicle-fixed *x*-axis
F_{yflt}, F_{yfrt}	Lateral force tire applied to front left and front right wheels, along the vehicle-fixed *y*-axis
F_{xrlt}, F_{xrrt}	Longitudinal tire force applied to rear left and rear right wheels, along the vehicle-fixed *x*-axis
F_{yrlt}, F_{yrrt}	Lateral force applied to rear left and rear right wheels, along the vehicle-fixed *y*-axis
F_{zf},F_{zr}	Normal force applied to front and rear wheels, along vehicle-fixed *z*-axis
F_{znom}	Nominal normal force applied to axles, along the vehicle-fixed *z*-axis
F_{zfl},F_{zfr}	Normal force applied to front left and right wheels, along vehicle-fixed *z*-axis
F_{zrl},F_{zrr}	Normal force applied to rear left and right wheels, along vehicle-fixed *z*-axis
M	Vehicle body mass
a, b	Distance of front and rear wheels, respectively, from the normal projection point of vehicle CG onto the common axle plane
H	Height of vehicle CG above the axle plane
D	Lateral distance from the geometric centerline to the center of mass along the vehicle-fixed *y*-axis
Hh	Height of the hitch above the axle plane along the vehicle-fixed *z*-axis
Dh	Longitudinal distance of the hitch from the normal projection point of tractor CG onto the common axle plane
Hl	Lateral distance from center of mass to hitch along the vehicle-fixed *y*-axis.
α_f, α_r	Front and rear wheel slip angles
α_{fl}, α_{fr}	Front left and right wheel slip angles
α_{rl}, α_{rr}	Rear left and right wheel slip angles

δ_f, δ_r	Front and rear wheel steering angles
δ_{rl}, δ_{rr}	Rear left and right wheel steering angles
δ_{fl}, δ_{fr}	Front left and right wheel steering angles
w_f, w_r	Front and rear track widths
Cy_f, Cy_r	Front and rear wheel cornering stiffness
Cy_{fdata}, Cy_{rdata}	Front and rear wheel cornering stiffness data
σ_f, σ_r	Front and rear wheel relaxation length
$\alpha_{f\sigma}$, $\alpha_{r\sigma}$	Front and rear wheel slip angles that include relaxation length
v_{wf}, v_{wr}	Magnitude of front and rear wheel hardpoint velocity
μ_f, μ_r	Front and rear wheel friction coefficient
μ_{fl}, μ_{fr}	Front left and right wheel friction coefficient
μ_{rl}, μ_{rr}	Rear left and right wheel friction coefficient
C_d	Air drag coefficient acting along vehicle-fixed *x*-axis
C_s	Air drag coefficient acting along vehicle-fixed *y*-axis
C_l	Air drag coefficient acting along vehicle-fixed *z*-axis
C_{rm}	Air drag roll moment acting about the vehicle-fixed *x*-axis
C_{pm}	Air drag pitch moment acting about the vehicle-fixed *y*-axis
C_{ym}	Air drag yaw moment acting about the vehicle-fixed *z*-axis
A_f	Frontal area

4.2. Driving Scenario

The Driving Scenario Designer is used to design synthetic driving scenarios. These scenarios are used to test the automatic driving systems. Create road and actor models using a drag-and-drop interface. Configure vision, radar, lidar, and INS sensors mounted on the ego vehicle. And it will be used to Import Road data from OpenStreetMap as shown in **Figure 4.2** and use this road as a reference path in Stanley controller.

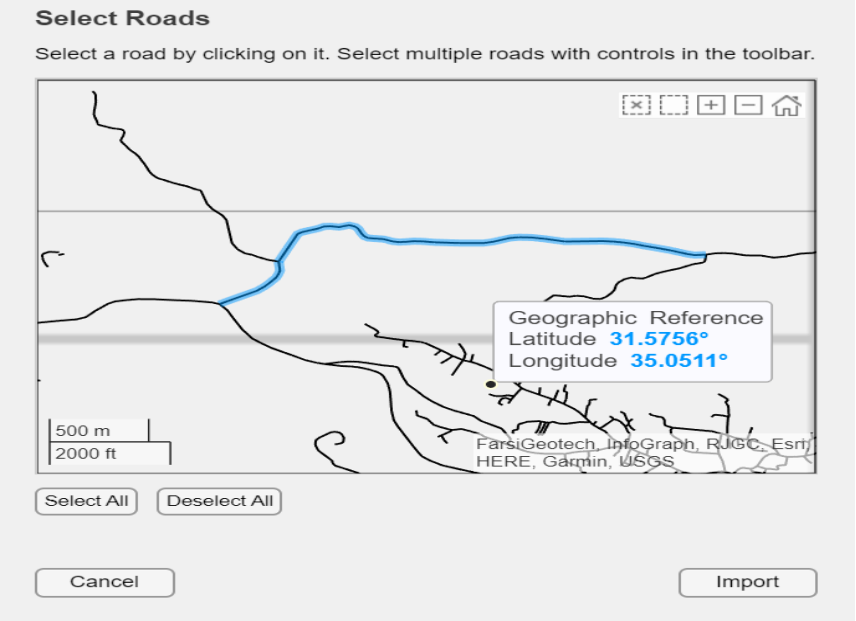


Figure 4.2. Road data from OpenStreetMap

4.3. connect Simulink Block:

We have reference data include path and the road data, that we export it from website *open street map*, and then import these data to driving scenario to get the Reference path. Then save the driving scenario file with form "driving scenario. Mat" then the file is been called in the model to calculate the desired steering angle for the path using lateral Stanley controller. And display it in the desired steering angle block as shown in the **Figure 4.4** below.

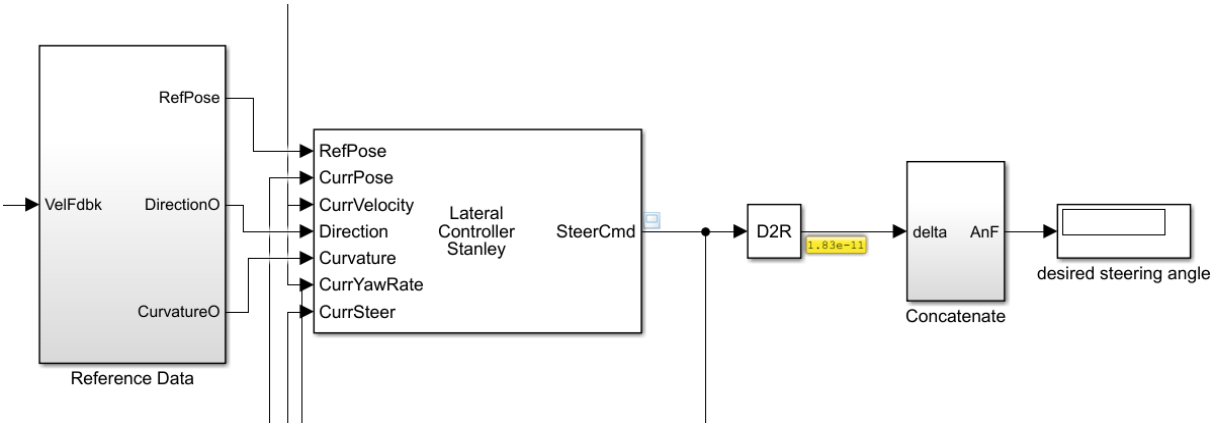


Figure 4.4. Blocks to calculate steering angle.

Then manually entering the value of the steering angle using kinematic steering block, and make it input in the pin “whlAngF” for the vehicle body 3DOF dual track block. As shown in the **Figure 4.5**.

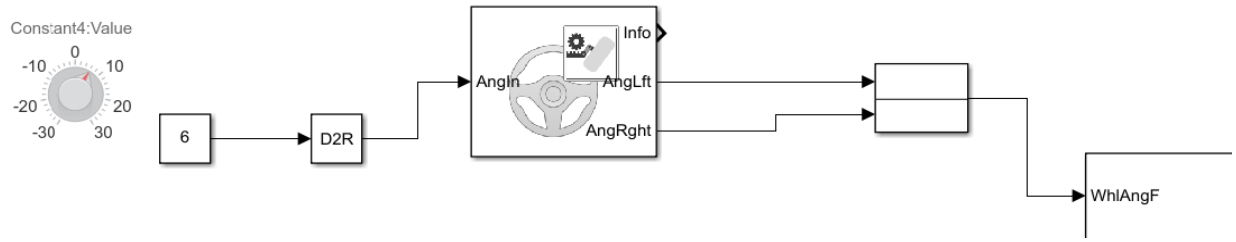


Figure 4.5. Employing a kinematic steering block to manually enter the steering angle's value..

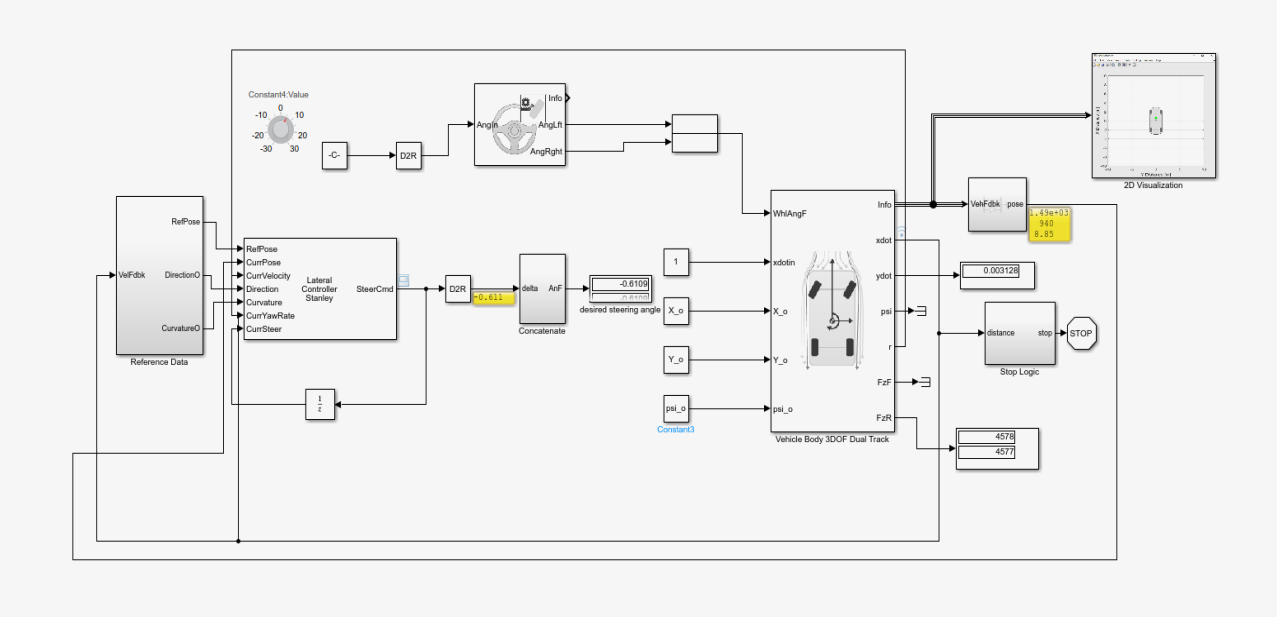


Figure 4.6. Vehicle Path Tracking Using Stanley Controller

After running the program Simulink, the result is been displayed on the 2D visualization screen.

5

Chapter 5

System Interface and Operations

Introduction

This chapter introduces module construction. It demonstrates how hardware parts are connected and interface systems. As well as how gather data and perform calculations and review the steps that were taken for the practical application of the project, starting from extracting information from the vehicle to presenting the results**.** Building on the previous steps that reviewed in the previous units**.**

5.1. Extract and Prepare ECU data

The experimental method depends on identifying, extracting, storing and visualizing the variables in order to determine the desired procedures to be followed. In this project, we used and experimented with four electronic pieces to extract and display data.

- **CAN Bus shield V2**
- **CAN Bus shield V1.2**
- **Sun-flower CAN Shield V2**
- **ELM327**

All of these pieces share one goal, which is to read CAN-BUS messages and specify the required information for model using PID**.**

In the beginning, specify the major steps before starting to use electronic parts **:**

1. Determine the type of vehicle that will be used in this project and selected **Mazda 3 2016 SKYACTIV-G 2.0 Automatic transaxle**

Figure 5.1. Mazda 3 skyactiv 2016

- **Specification**

Table 5.1. Specifications for mazda3-G skyactive

Parameter	Vehicle specification		
Overall length	4,580 mm (180.3 in)		
Overall width	1,795 mm (70.7 in)		
Overall height	1,455 mm (57.3 in)		
Front tread Lf	1,555 mm (61.2 in)		
Rear tread Lr	1,560 mm (61.4 in)		
Wheelbase L	2,700 mm (106.3 in)		
Weights	1,815 kg (4,001 lbs)	Front	975 kg (2,149 lbs)
		Rear	848 kg (1,870 lbs)
Tire size	215/45R18 89W	Front	250 kPa (36 psi)
		Rear	250 kPa (36 psi)
Front Cornering Stiffness	56 kN/rad		
Rear Cornering Stiffness	65 kN/rad		
Centre of Gravity Height	0.533 m		
Wind Force Centre Height	0.5 m		
Air Drag Coefficient	0.31		

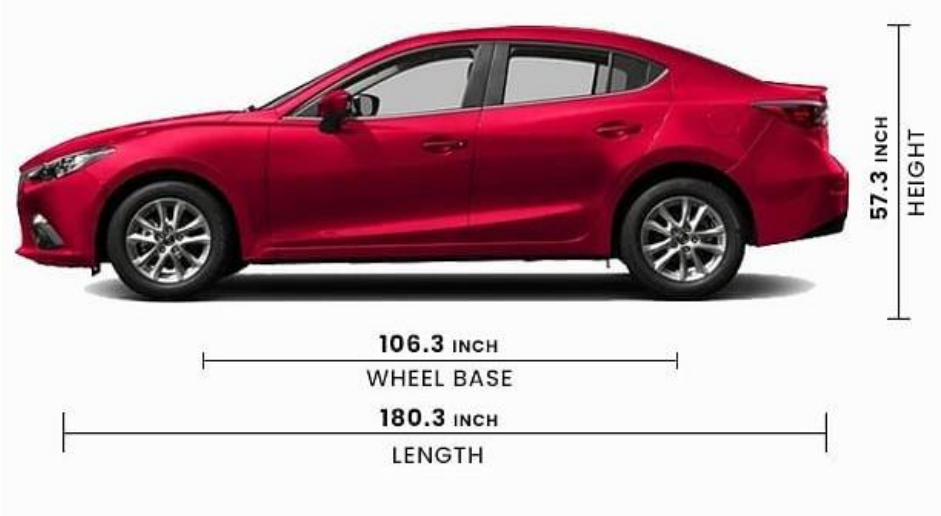


Figure 5.2. The vehicle dimension

2. OBD II

On-board diagnostics (OBD, ISO 15765) is a self-diagnostic and reporting capability that e.g. mechanics use to identify car issues. OBD2 specifies diagnostic trouble codes (DTCs) and real-time data (e.g. speed, RPM), which can be recorded via OBD2 loggers.

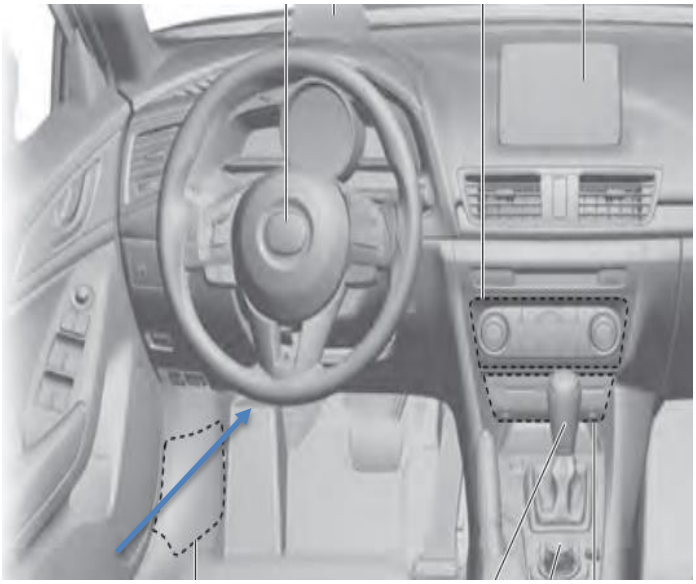


Figure 5.3 .OBD II connector location for Mazda 3

3. Defined CAN high CAN low pin

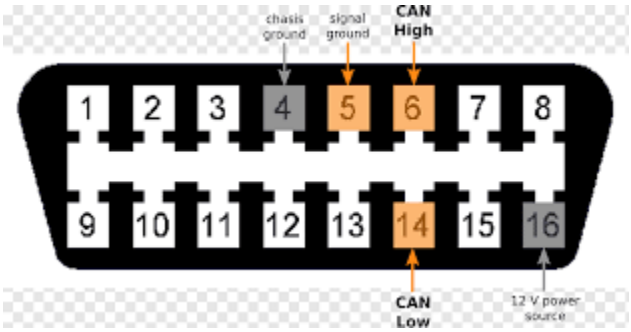


Figure 5.4 . CAN high CAN low pin

4. Checking CAN Voltage

1. Disconnect all devices except for the device being tested, then power the device on.
2. Measure voltage on any of disconnected plugs between CAN HI and GROUND. The resulting voltage should be between 2.5 and 3.0VDC.
3. At the same location, measure voltage between CAN LOW and GROUND. The resulting voltage should be between 2.5 and 2.0 VDC.

A low voltage of 1.4VDC or less on either of these indicates a potential failure on the CAN port of the device.

If voltages are exactly 2.50 VDC, and do not change after several seconds, this indicates the device connected is powered but not broadcasting data

5. CAN bus testing with an oscilloscope

1. Connect the oscilloscope to CAN bus with a probe.
    - Connect channel 1 with CAN high and CAN low with channel 2 from OBD II

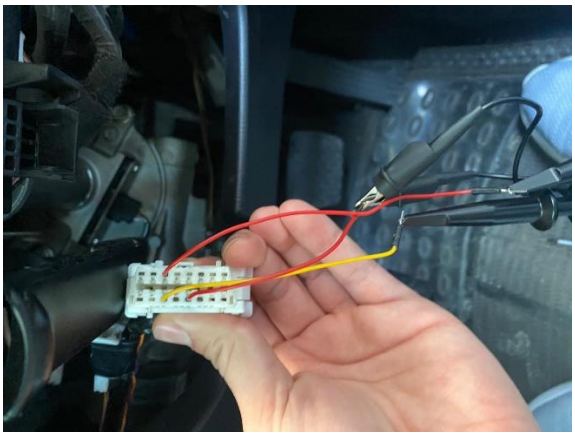


Figure 5.5 .OBD II connector location for Mazda 3

2. Determine the scale of the signal and adjust your trigger. The simplest and fastest way to do this is to press "Auto Scale"
3. Turn on "Serial" on the front panel of the oscilloscope

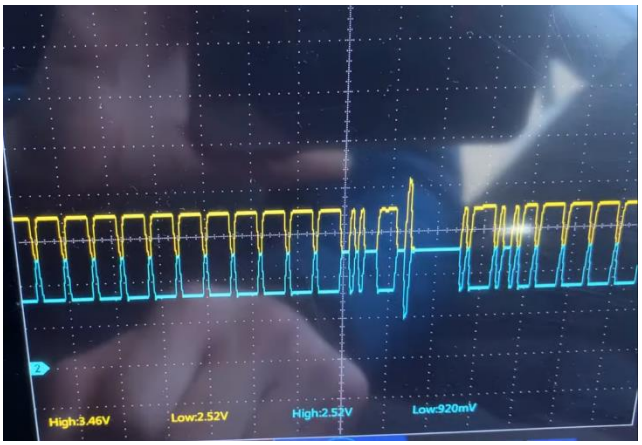
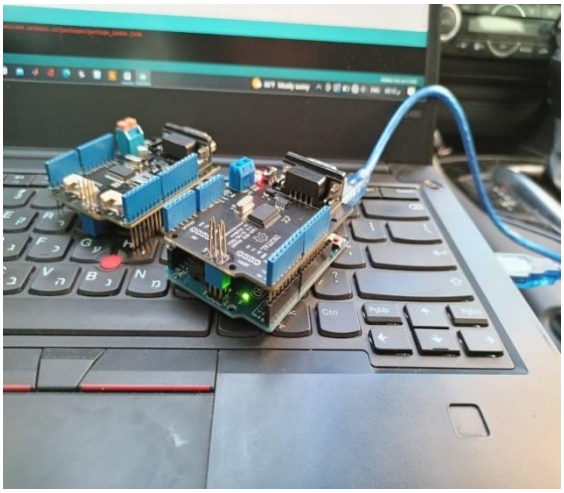


Figure 5.6 . signal CAN high and CAN low

Figure 5.7 . Micsig Tablet Oscilloscope Serial

5.1.1 CAN Bus Shield V2

The CAN Bus shield was interfaced with an Arduino Uno in accordance with the data sheet. There are a number of variables that need to be calibrated with the CAN Bus shield, Arduino, and can bus in the vehicle, such as choosing the SPI pin, followed by the can bus speed and author parameter, as shown in **Figure 5.8**. Then, using an OBD to DB9 connection, they were linked to the car's OBD and prepared and uploaded with the Arduino code "receive check" the code will be attached in **appendix F"** to read the messages that would be received from the CAN bus, as shown in **Figure 5.9.**



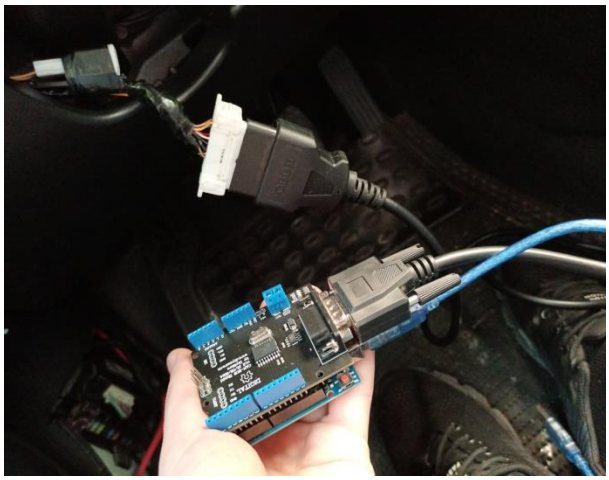


Figure 5.8 CAN bus shield with Arduino Uno

Figure 5.9 Interfacing with OBD of the car

As seen in Figure 5.10, no massage was given following that tune on the car. As a result, when trying again with a different code that is "CAN read," the same "no data" results were produced.

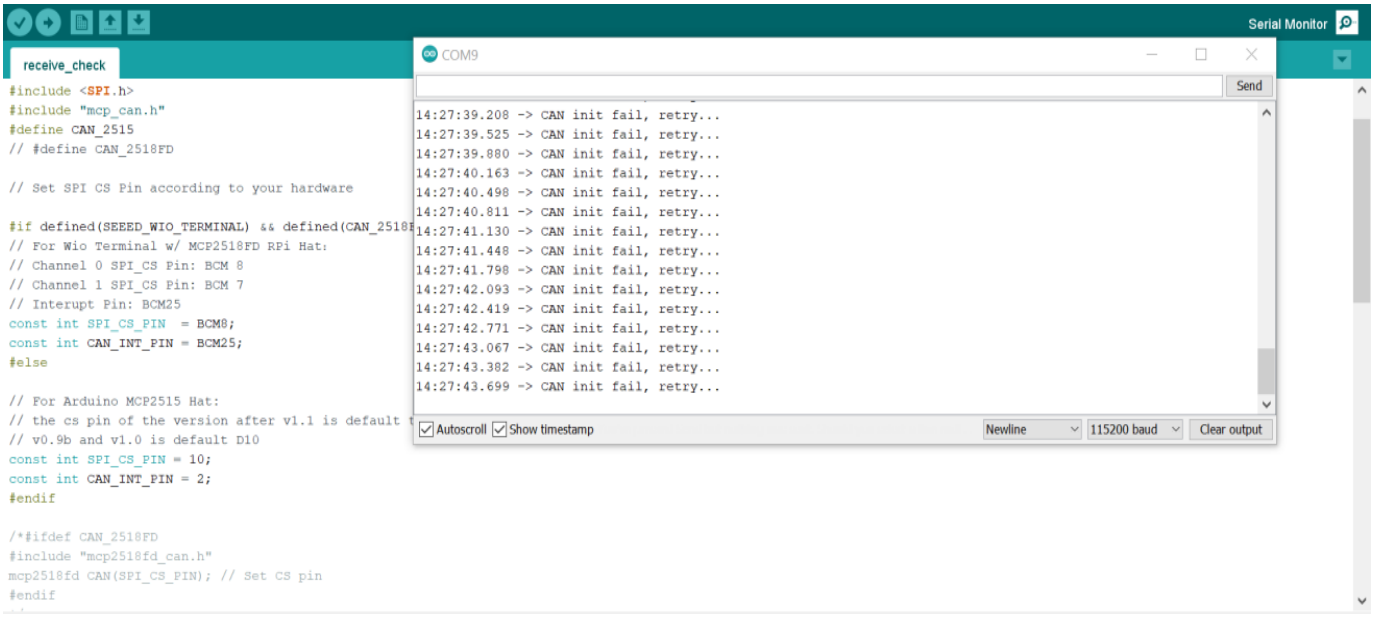


Figure 5.10 Serial monitor in Arduino

Using inspection tools like the launch device, it has been verified the car's CAN bus's functionality. To confirm the connections and components, and also checke the voltage and resistance between the CAN-H and CAN-L lines.

5.1.1.1 CAN Bus Shield V1.2

The CAN bus shield V1.2 is the same as the V2 but in a different version; it has been paired with Arduino and the car OBD. We read the messages in the serial monitor after uploading the code to the shield, as shown in **Figure 5.11.**

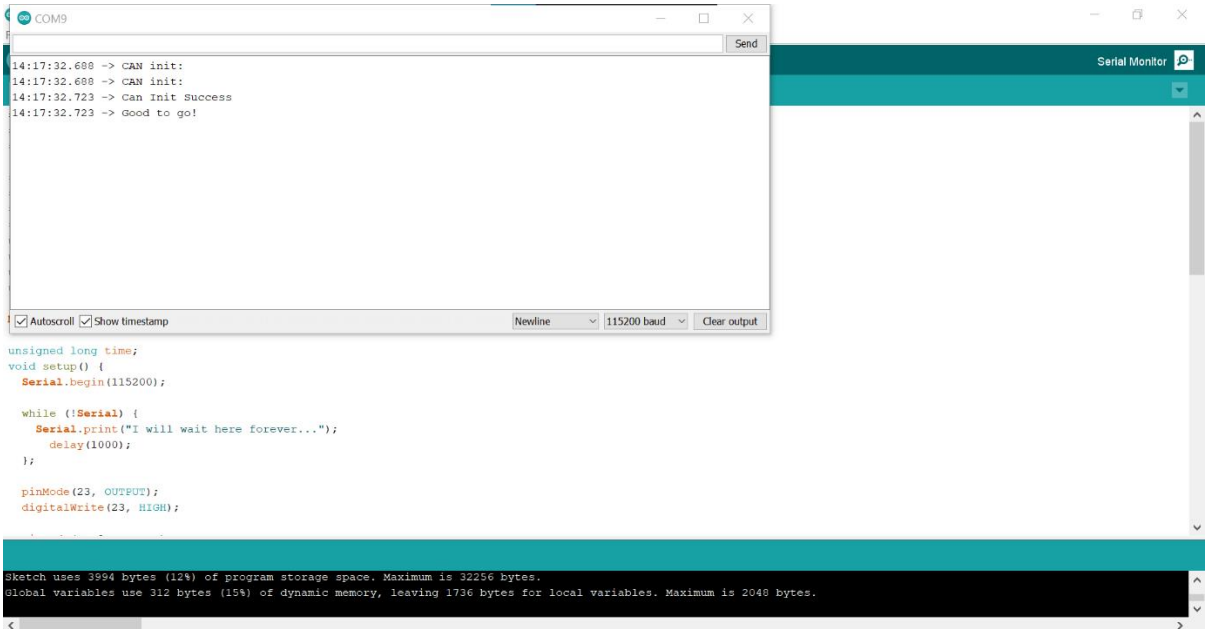


Figure 5.11. Serial monitors in Arduino CANBUS shield v1.2

The communication with the vehicle's CAN bus is success at this point, but couldn't able to determine the appropriate mode to read the data from the vehicle because there is no enough information to determine the appropriate mode.

5.1.2 Sunflower Shield

Because there is no data sheet for the sunflower shield or any reference site to obtain information about it, we encountered a problem dealing with it and setting the appropriate mode of operation. So, a several steps have been token to verify the shield's pins and the type of processor used in order to understand its characteristics and how to deal with it.

The sunflower shield is interfaced with an Arduino Uno in a suitable way, as demonstrated in **Figure 5.12.a** Then, prepared and uploaded the Arduino **code "CAN read" "the code will be attached in appendix G"** to read the messages that the car's CAN bus will send, and then linked it to the OBD of the vehicle using an OBD to DB9 connector, as shown in **Figure 5.12.b**

Figure 5.12a Sun-flower shield with an Arduino Uno cable

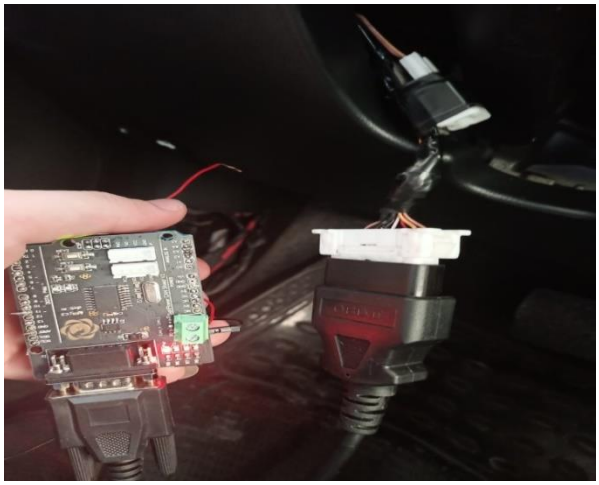


Figure 5.12b Sun-flower with OBD to DP9

After that tune on the car and no massage was received as shown in **Figure 5.13.**

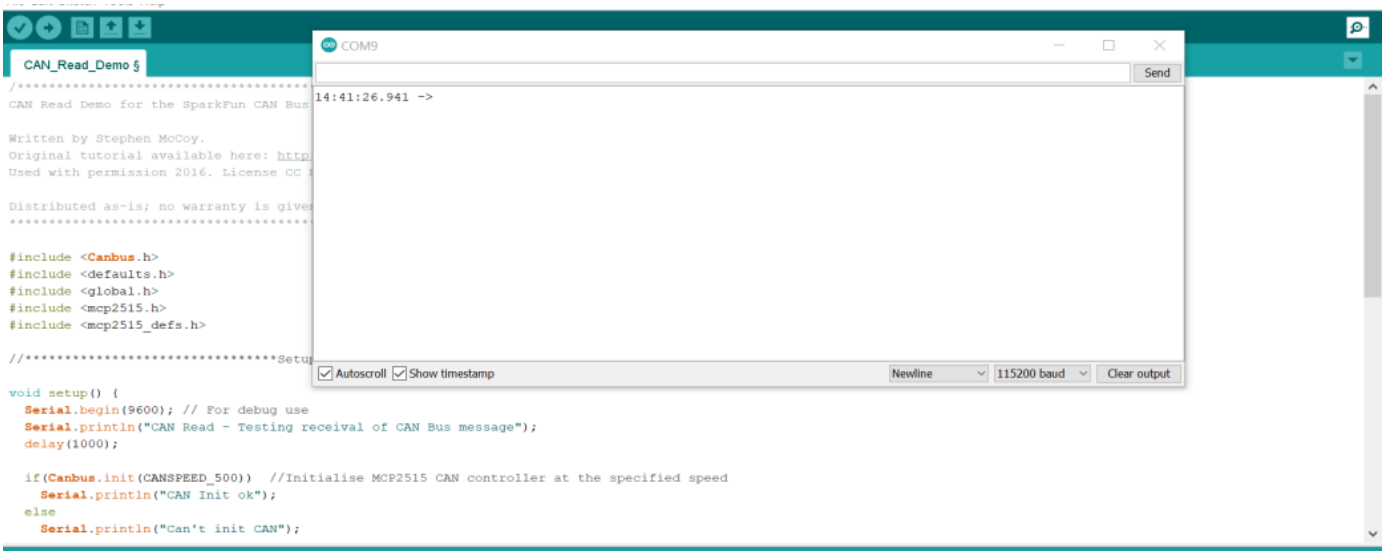


Figure 5.13. Serial monitor in Arduino

5.1.3 ELM and Python

Hardware implementation

We plugged ELM327 with OBD port then connect built-in laptop Bluetooth with ELM327 Bluetooth adapter to communicate.

Software implementation

Install the last version of python-OBD library from pypi using pip install then Import python OBD in python script, after that use the python-OBD library to connect to an ELM327 via a Bluetooth port in Python To get information from the car, we send commands that query for the required data(such RPM and vehicle speed).

But about steering angle sensor data is not available, because the PIDs of steering angle is not implemented in stander in ISO 15765 in another ward is not general.

Then by refer to service manual the data available for the sensor is:

Purpose, Function [13]

- The steering angle sensor outputs the steering angle and steering angle reference point during the period which the EPS control module performs initial learning.

Construction

- The steering angle sensor is installed to the clock spring.
- Consists of the gear which rotates together with the steering (magnet) and the magnetic sensor

Operation

- By the rotation of the gear (magnet) corresponding to the steering operation, the magnetic sensor output value changes.
- The steering angle sensor outputs the magnetic sensor output to the start stop unit.

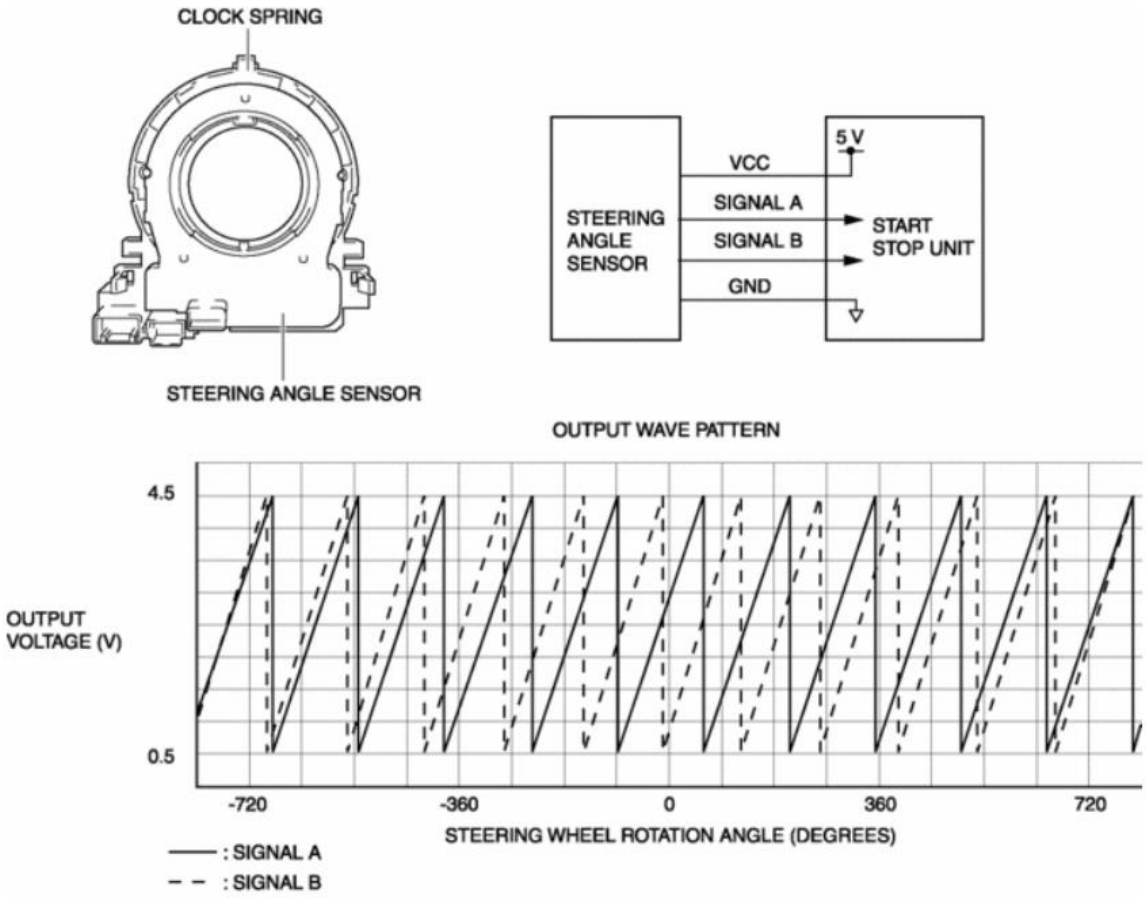


Figure 5.14 streering angle sensor signal [13]

But when we display the two signals by myDAQ (data acquisition tool) the signal is different from the desired output like we see in **figure 5.15**

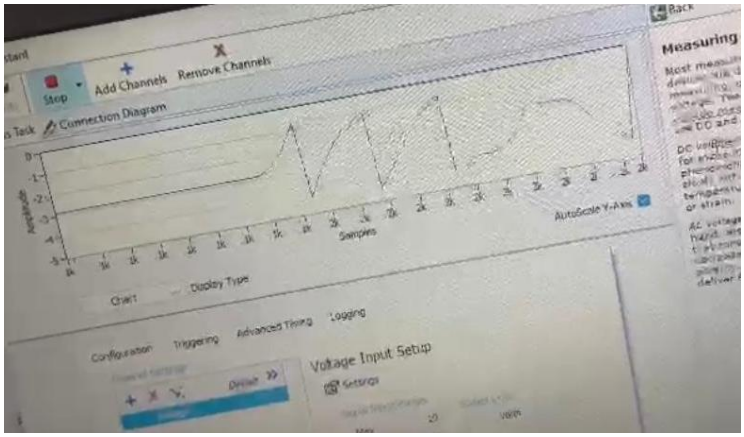


Figure 5.15.a the signal A from sensor steering angle by myDAQ

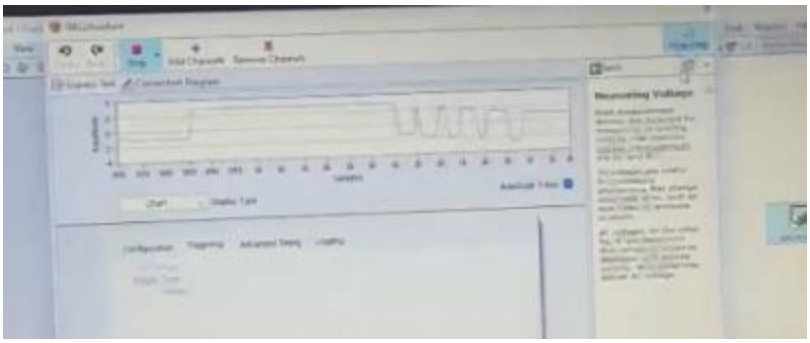


Figure 5.15.b The signal B from sensor steering angle by myDAQ

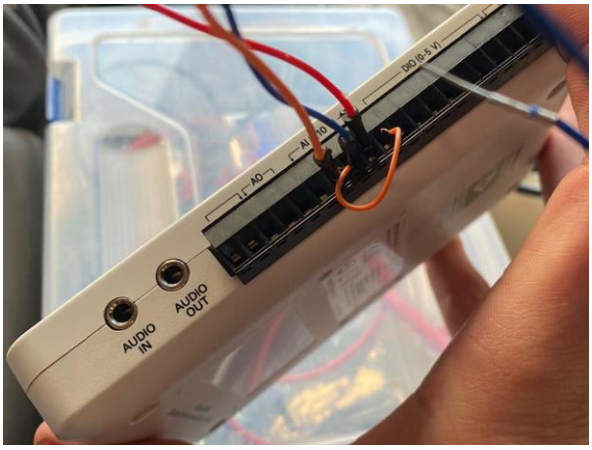


Figure 5.15.C myDAQ

Implementation issues:

The latest Python OBD release was released in 2019 and doesn't work in Python 3.10 and 3.11, so, to solve this problem an older version of Python is installing which is (3.8.4).

5.2 Using a multi-turn potentiometer to extract steering angle

After we were unable to extract the steering angle reading from the electronic control unit ECU and read the sensor signal itself, we took a third procedure by connecting the steering wheel shaft with a potentiometer multiturn , so design a cylinder that put on a shaft that moves the potentiometer according to the rotation of the steering wheel in the next figure explain the new Block Diagram **Figure 5.16**

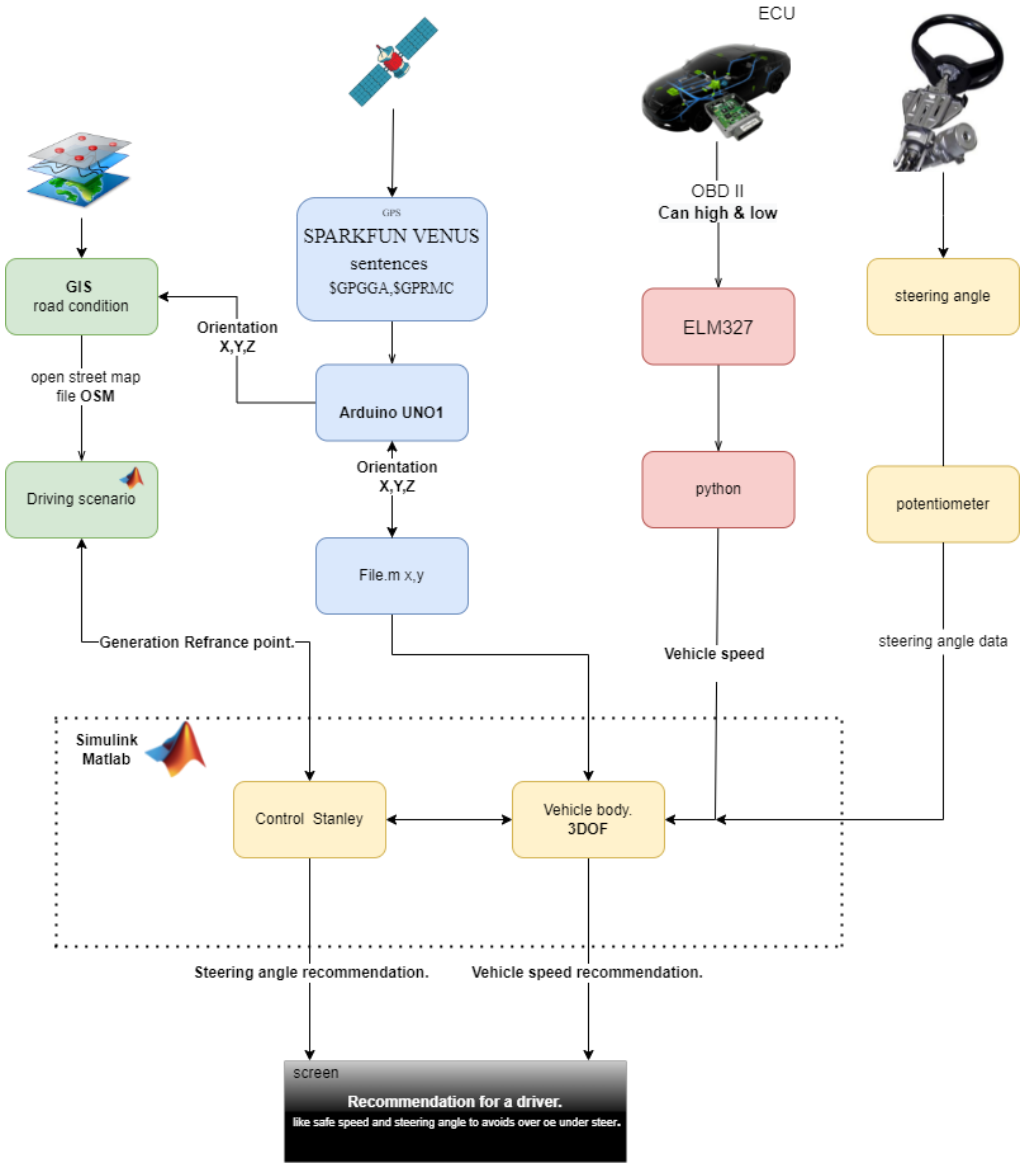


Figure 5.16 block diagram for our project procedure

Determine the transmission ratio between the steering shaft and the potentiometer which is 1 to 2, and Make a design for printing a cylinder using CATIA software **figure 5.17**

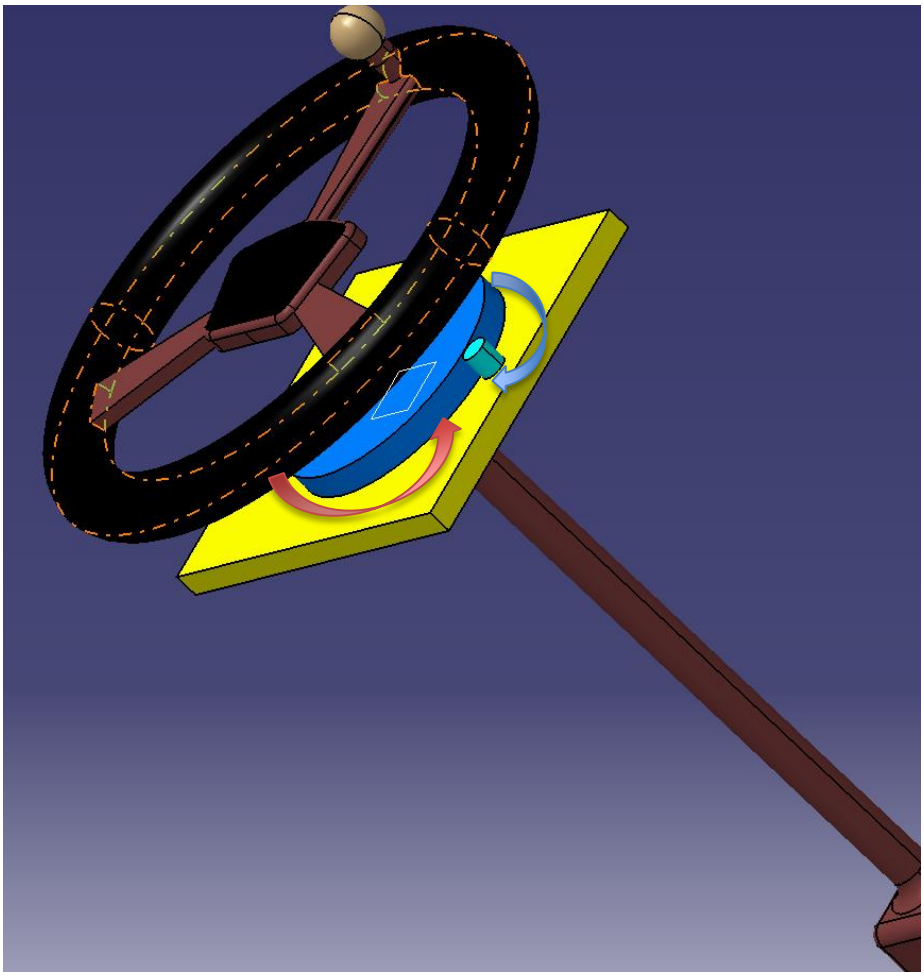


Figure 5.17 Design for steering wheel with potontiometer sensor

After printing the two cylinders, installed a Toothed belt on the two cylinders and installed it on the steering shaft **Figure 5.18**

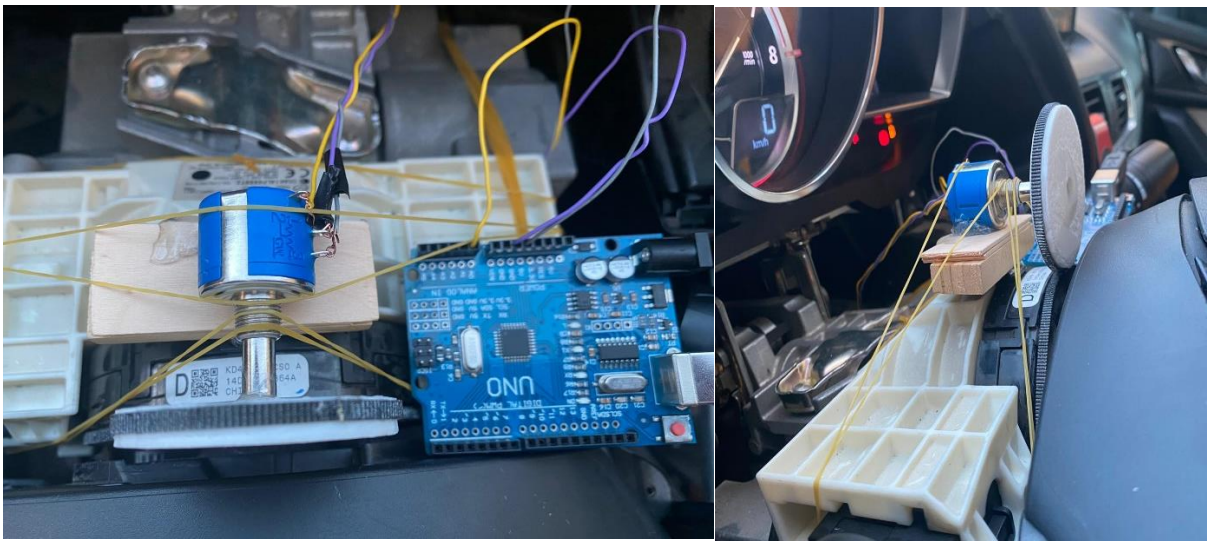


Figure 5.18 Tools steering angle measurement

After installing the potentiometer, write the special Arduino code **"the code will be attached in appendix I"** in the potentiometer and map the readings and calibrate it. After calibrating it, and verified the readings using the lunch device **figure 5.19**

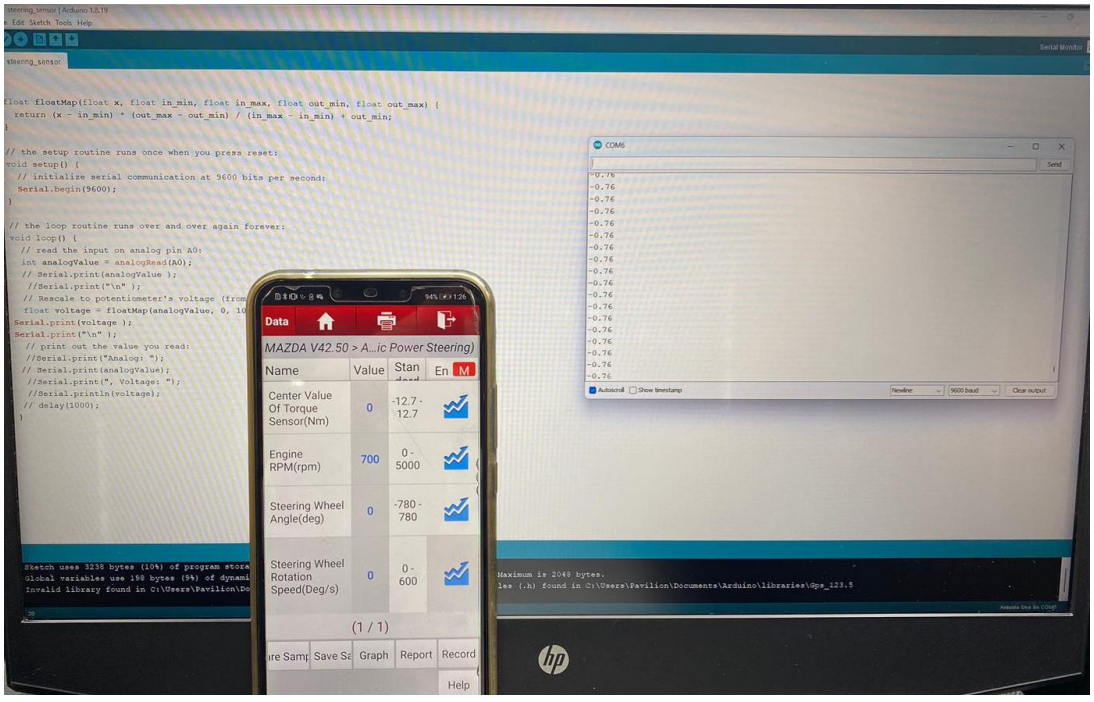


Figure 5.19 Verified the readings using the lunch device

5.3 Receiving and Preparing GPS data

5.3.1 GPS with Arduino connection

The second step in implementing our system was the creation of a GPS control module (GPS receiver) that permits to receive GPS data (Long and Lati) from satellite. Two crucial components, the microcontroller (Arduino Uno) and the GPS sensor, were utilized for this purpose. As soon as this connection is established and power is supplied to the Arduino, the GPS sensor will begin immediately receiving data. These component connections are explained **in the figure 5.20 and table 5.2** below.

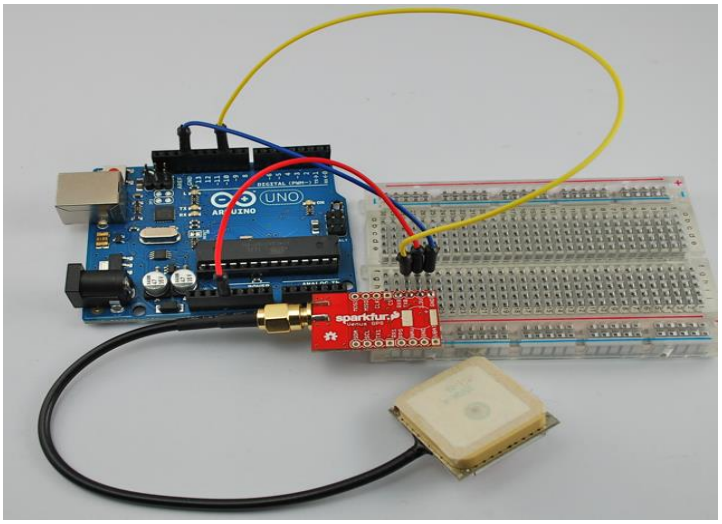


Figure 5.20 GPS sensor connection

Table 5.2 GPS sensor and Arduino Uno pins

Arduino	GPS sensor
V_{cc}	V_{cc}
Ground	Ground
Digital pin 10	T_x

5.3.2 Convert Longitude and latitude to X,Y,Z coordinate

In order to convert GPS data ((Long and Lat) to X,Y,Z coordinate, by using equations in python: using python code so read data from GPS sensor from Arduino and send longitude and latitude as permitters for a the function long in python start read data and save data in file text file.

After all of this need convert text file to MATLAB file (m.file) and from driving scenario to generate reference point.

5.4 Preparing GIS data

At the beginning of work on the project, we took and drew an external path in the Halhul area, but due to the difficulty of leaving campus university boundary because the vehicle an illegal that was taken from the police, we drew a path inside the university from Building B to Building C. so determined the path to be taken using the university campus AutoCAD drawings **Figure 5.21** and taking the coordinates of the track using the GPS sensor and taking a file containing the route information in the OSM file format and inserting it into the MATLAB Driving Scenario to analyze the file.

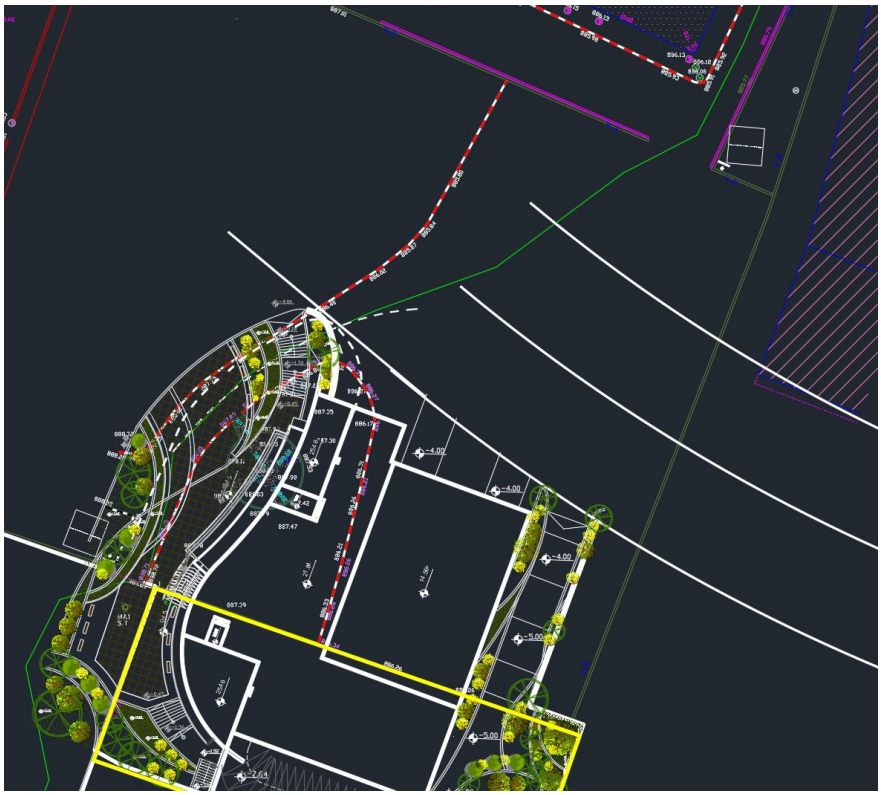


Figure 5.21 AutoCAD drawing for Campus

After that, created the path inside the university campus on the OpenStreetMap website and uploaded all the characteristics and information of the path (the length of the path, its Cartesian coordinates, the type of road, the width of the road, the specified speed, etc.) on the site, for approval by the site **Figure 5.22**

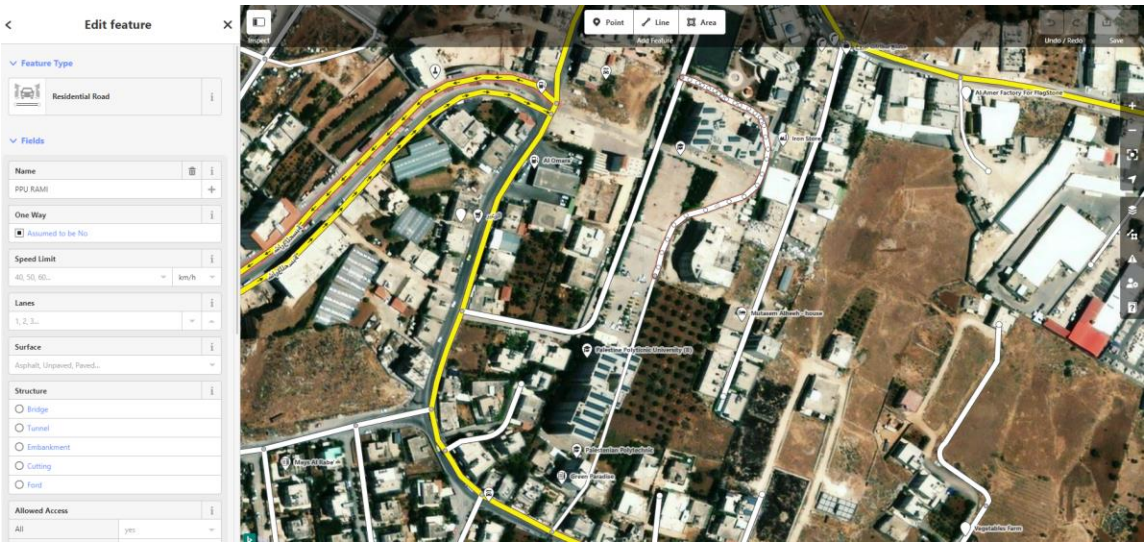
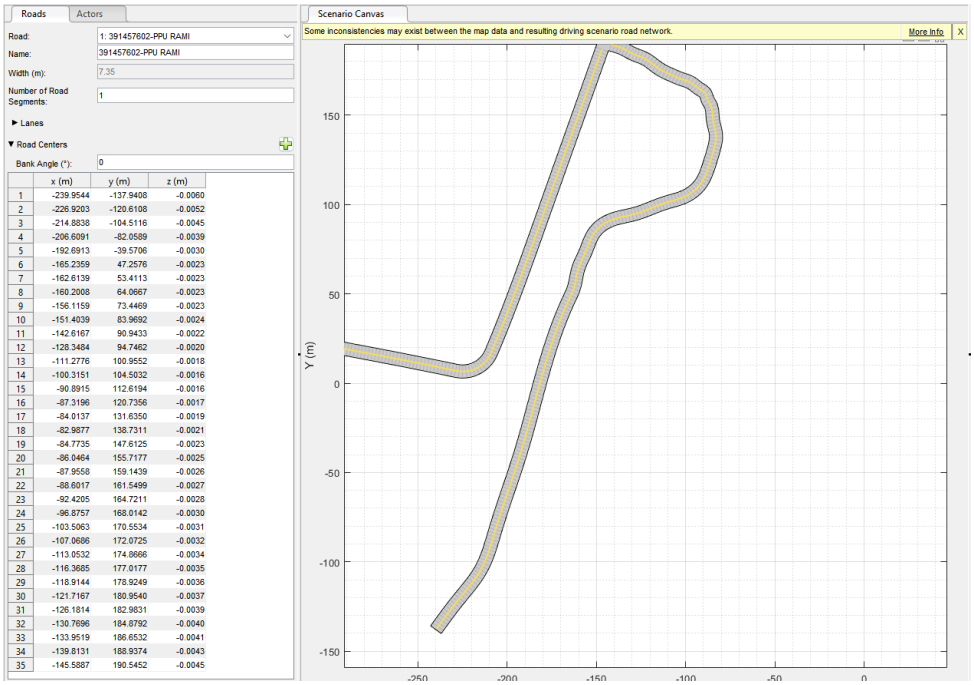


Figure 5.22 Open-street map drawing for Campus

After that, extract the road data in an OSM file (GIS File) This is to import it to MATLAB and use it in the Driving Scenario application **Figure 5.23**.



	x (m)	y (m)	z (m)
1	-239.9544	-137.9408	-0.0060
2	-226.9203	-120.6108	-0.0052
3	-214.8838	-104.5116	-0.0045
4	-206.6091	-82.0589	-0.0039
5	-192.6913	-39.5706	-0.0030
6	-165.2359	47.2576	-0.0023
7	-162.6139	53.4113	-0.0023
8	-160.2008	64.0667	-0.0023
9	-156.1159	73.4469	-0.0023
10	-151.4039	83.9692	-0.0024
11	-142.6167	90.9433	-0.0022
12	-128.3484	94.7462	-0.0020
13	-111.2776	100.9552	-0.0018
14	-100.3151	104.5032	-0.0016
15	-90.8915	112.6194	-0.0016
16	-87.3196	120.7356	-0.0017
17	-84.0137	131.6350	-0.0019
18	-82.9877	138.7311	-0.0021
19	-84.7735	147.6125	-0.0023
20	-86.0464	155.7177	-0.0025
21	-87.9558	159.1439	-0.0026
22	-88.6017	161.5499	-0.0027
23	-92.4205	164.7211	-0.0028
24	-96.8757	168.0142	-0.0030
25	-103.5063	170.5534	-0.0031
26	-107.0686	172.0725	-0.0032
27	-113.0532	174.8666	-0.0034
28	-116.3685	177.0177	-0.0035
29	-118.9144	178.9249	-0.0036
30	-121.7167	180.9540	-0.0037
31	-126.1814	182.9831	-0.0039
32	-130.7696	184.8792	-0.0040
33	-133.9519	186.6532	-0.0041
34	-139.8131	188.9374	-0.0043
35	-145.5887	190.5452	-0.0045

Figure 5.23 Uplode GIS file in Driving Scenario application

5.5 Data linking with MATLAB Software

First, prepare ECU data in m.file format for transfer to MATLB, including vehicle speed. Then gather data from the steering angle sensor and send it via Arduino to the MATLAB software. After that, take the previously prepared GIS file and send it to MATLAB (driving scenario). **Figure 5.24**

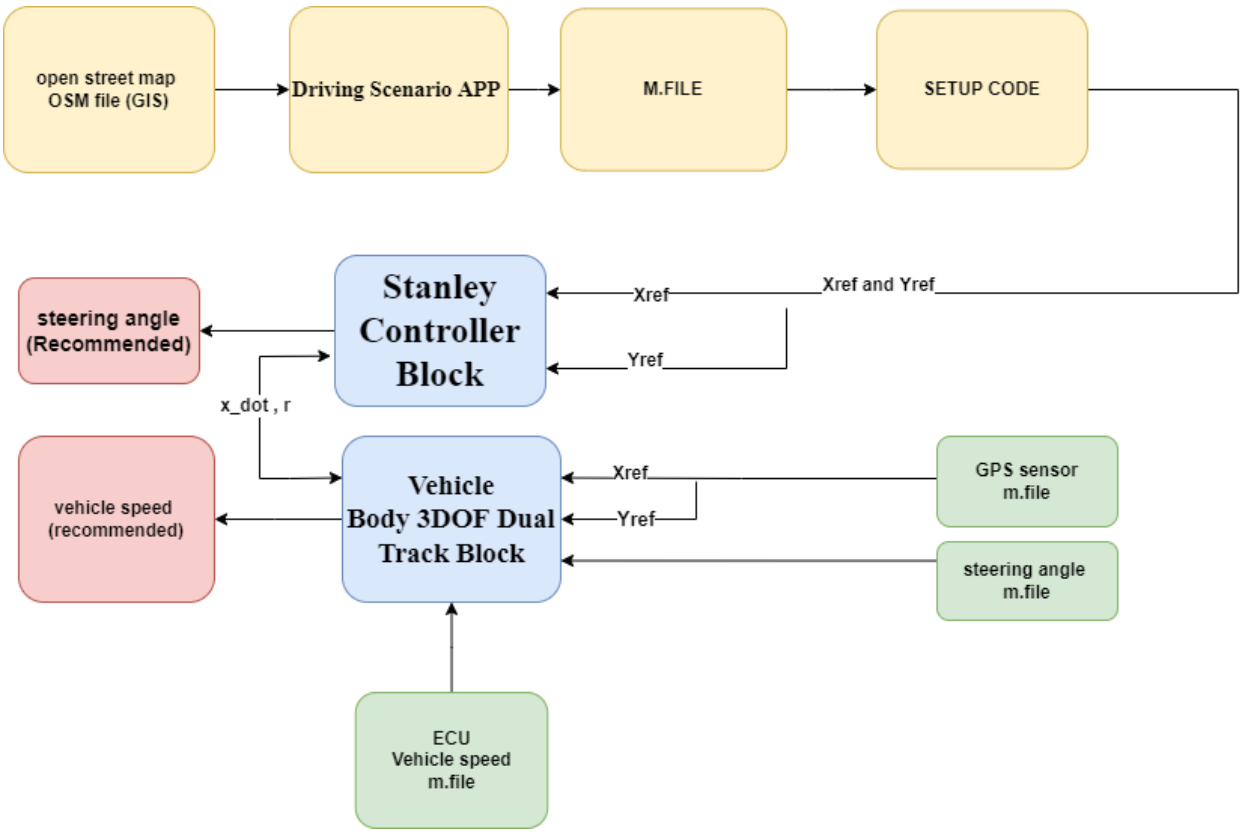


Figure 5.24 Interfacing data with MATLAB

6

Chapter 6

Experimentation and Results

Introduction

The experiment test of any system is seen to be one of the most important step in system development since it allows the researcher to assess whether or not their work was done correctly. On the other hand, the experiment test is thought to be the primary source of system feedback.

In addition to the experimental approach, results, and suggestions in this chapter, which also discusses the state-of-the-art of this technology, there are several key considerations that should be kept in mind when carrying out the experimental test.

6.1 Experimentation

6.1.1 Result for GPS and GIS test data

As we explained in the previous units, we need information about the location of the vehicle and the information about the road. With the help of Sensor GPS, took the vehicle in a path and read the longitudinal and latitude **table 6.1.**

1 latitude	2 longitude	3 altitude
31.5075	35.0908	891.1270
31.5076	35.0908	891.5040
31.5076	35.0908	891.3730
31.5076	35.0908	891.2970
31.5076	35.0908	891.6690
31.5076	35.0908	891.0460
31.5076	35.0908	891.5810
31.5076	35.0908	891.1210
31.5076	35.0908	891.3360
31.5076	35.0908	891.6410
31.5076	35.0908	891.4090
31.5076	35.0908	892.0790
31.5076	35.0907	891.4430
31.5076	35.0907	892.1710
31.5076	35.0907	891.4770
31.5076	35.0907	891.7600
31.5076	35.0907	891.8450
31.5076	35.0907	891.8620
31.5076	35.0907	892.0550
31.5076	35.0907	891.6290
31.5076	35.0907	891.8330
31.5076	35.0907	892.2720
31.5076	35.0907	892.5480
31.5076	35.0907	892.0900
31.5077	35.0906	892.5090
31.5077	35.0906	892.3610

Table 6.1. Result for GPS in longitude and latitude >>> above the table ??!!!

- **Accuracy:** The NMEA protocol includes a sentence (GPRMC) that reports the current accuracy of the GPS receiver's position. You could report this accuracy in meters or other units of distance. For example, you could report that the average accuracy was **0.53**meters, or that the maximum accuracy was **1** meter.

❖ Result for converting WGS 84 Format into X, Y, Z coordinate (python)

X	Y
4456074.49	3126503.786
4456074.546	3126502.939
4456074.872	3126502.283
4456075.018	3126501.499
4456075.344	3126500.843
4456075.462	3126500.482
4456075.698	3126499.762
4456076.024	3126499.106
4456076.44	3126498.513
4456076.378	3126498.026
4456076.795	3126497.433
4456077.328	3126496.48
4456077.357	3126496.056
4456077.592	3126495.336
4456077.71	3126494.976
4456077.829	3126494.616
4456077.765	3126494.13
4456078.002	3126493.41
4456078.21	3126493.113
4456078.147	3126492.626
4456078.473	3126491.97
4456078.411	3126491.484
4456078.827	3126490.89
4456078.737	3126490.827

Table 6.2. Result for GPS in X and Y >>> above the table ??!!!!

6.1.2 Result for extract data from ECU

We chose a way to test ELM327 python code to collect real time data for the speed of the vehicle and RPM. And we noticed that in some readings it gives NONE VALUE. So, we solved this problem by substituting the previous value for the NONE VALUES. **Figure 6.1**

```
15.0 kph    652.25 revolutions_per_minute time :1669810729.642763
None  642.5 revolutions_per_minute  time :1669810730.0296946
14.0 kph    None  time :1669810730.3482237
13.0 kph    650.75 revolutions_per_minute time :1669810730.6453242
13.0 kph    642.0 revolutions_per_minute  time :1669810730.876307
12.0 kph    None  time :1669810731.2793713
12.0 kph    638.0 revolutions_per_minute  time :1669810731.5554883
11.0 kph    637.25 revolutions_per_minute time :1669810731.8762996
10.0 kph    635.5 revolutions_per_minute  time :1669810732.2853055
None  None  time :1669810732.60745
None  None  time :1669810732.925203
8.0 kph     646.0 revolutions_per_minute  time :1669810733.323826
8.0 kph     None  time :1669810733.7430434
None  None  time :1669810734.1307461
8.0 kph     None  time :1669810734.8568788
8.0 kph     659.75 revolutions_per_minute time :1669810735.1846423
8.0 kph     669.0 revolutions_per_minute  time :1669810735.5554252
None  667.75 revolutions_per_minute time :1669810735.952807
9.0 kph     None  time :1669810736.2831364
8.0 kph     642.0 revolutions_per_minute  time :1669810736.6127005
7.0 kph     646.0 revolutions_per_minute  time :1669810737.093081
6.0 kph     652.75 revolutions_per_minute time :1669810737.4946108
5.0 kph     654.0 revolutions_per_minute  time :1669810737.9055734
4.0 kph     656.0 revolutions_per_minute  time :1669810738.334595
4.0 kph     662.5 revolutions_per_minute  time :1669810738.7360778
```

Figure 6.1 Data from ECU RPM and Vehicle speed

There are two possibilities for the reason

1. First possibility is that there is a momentary interruption of the Bluetooth connection during the experiment
2. The second possibility is due to the very large transfer speed and also the presence of priority messages.

After that, draw the results using the MATLAB program as show in **Figure 6.2**

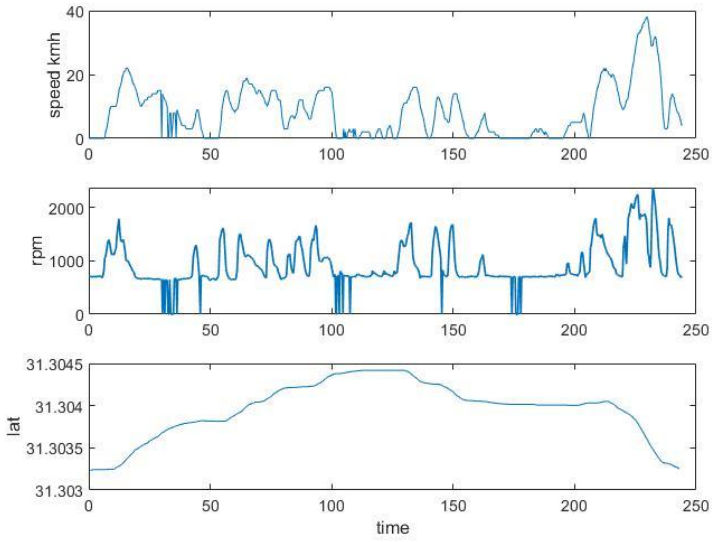


Figure 6.2 Data from ECU RPM and Vehicle speed MATLAB

6.1.3 Result for steering angle

In this section, we review the extracted results of the steering angle that were recorded while taking the readings of the track experiment within the university campus in as show in **Figure 6.3**.

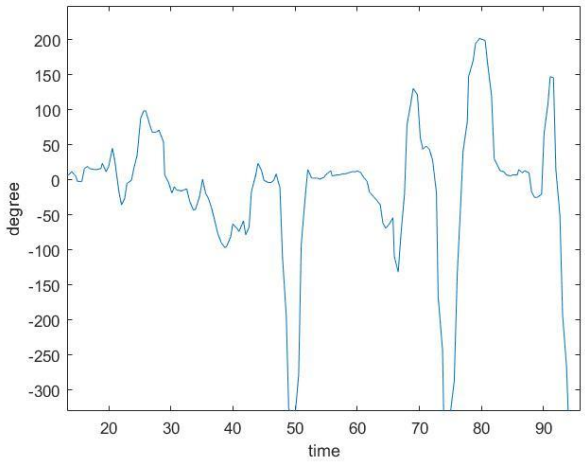


Figure 6.3 Steering angle result

All the readings extracted from the experiment we took for the track **Table 6.3 the vehicle speed , RPM , the time of take the information from vehicle , GPS data (longitudes ,latitude) every Colum has a name and unit .**

	A speedM_s Number	B timeCpu Number	C time0 Number	D timeCpuSpeed Number	E lat Number	F long Number	G degree Number	H VarName8 Text	I timeCpuV Number	J speedKm_h Number
21	3.0556	1.6722e+09	10.5910	1.6722e+09	31.5056	35.0901	6.8600		5.2216	11
22	3.3333	1.6722e+09	10.9653	1.6722e+09	31.5057	35.0901	5.3400		5.5267	12
23	3.6111	1.6722e+09	11.5886	1.6722e+09	31.5057	35.0901	2.2900		5.7399	13
24	3.8889	1.6722e+09	11.9861	1.6722e+09	31.5058	35.0901	-5.3400		5.9681	14
25	3.8889	1.6722e+09	12.7139	1.6722e+09	31.5058	35.0901	-5.3400		6.2536	14
26	4.1667	1.6722e+09	12.9226	1.6722e+09	31.5059	35.0902	5.3400		6.4579	15
27	4.4444	1.6722e+09	13.5263	1.6722e+09	31.5059	35.0902	6.8600		6.6604	16
28	4.4444	1.6722e+09	14.0499	1.6722e+09	31.5060	35.0902	11.4400		6.8877	16
29	4.7222	1.6722e+09	14.6832	1.6722e+09	31.5060	35.0902	5.3400		7.1685	17
30	5	1.6722e+09	14.9642	1.6722e+09	31.5061	35.0903	-2.2900		7.3698	18
31	5.5556	1.6722e+09	15.6774	1.6722e+09	31.5061	35.0903	-2.2900		7.5750	20
32	5.5556	1.6722e+09	16.0812	1.6722e+09	31.5062	35.0903	16.0100		7.7782	20
33	6.3889	1.6722e+09	16.5849	1.6722e+09	31.5062	35.0903	19.0600		7.9978	23
34	6.6667	1.6722e+09	16.9345	1.6722e+09	31.5062	35.0903	16.0100		8.4114	24
35	6.9444	1.6722e+09	17.6537	1.6722e+09	31.5062	35.0903	14.4900		8.5896	25
36	7.2222	1.6722e+09	18.0630	1.6722e+09	31.5063	35.0904	14.4900		8.8105	26
37	7.5000	1.6722e+09	18.7475	1.6722e+09	31.5063	35.0904	16.0100		9.0041	27

gps2

Table 6.3 Result for All data from experiment (speed ,RPM ,time,…etc.)

6.2 Results

When it comes to vehicle navigation and control, understanding the appropriate speed and steering angle of a vehicle is essential. This is why we decided to use MATLAB to extract this information through a program consisting of vehicle dynamics equations, geographic information systems (GIS), and global positioning system (GPS). The vehicle dynamics equations allow to calculate the motion of a vehicle as it moves through space, taking into account its mass, acceleration, speed, and trajectory. By connecting these equations to geographic information systems (GIS), then map the vehicle's path and create a visual representation. This can then be combined with GPS to accurately pinpoint the vehicle's current position and its desired destination. The result of this program is the ability to accurately extract the appropriate speed and steering angle for a given vehicle. This is invaluable for any kind of navigation or vehicle control, as it allows to precisely control the vehicle's path without having to manually adjust its speed or angle. Overall, the combination of MATLAB, vehicle dynamics equations, GIS, and GPS has proven to be a powerful solution for extracting the appropriate speed and steering angle of a vehicle. It is a reliable and efficient way of controlling the vehicle's path and ensuring that it stays on course.

Through the experiment on the path used, the results is previously clarified, and insert this data (the speed of the vehicle and the steering angle and GPS and GIS information) to the model used in the program MATLAB Simulink, is explained in **chapter 4**. The results were extracted (vehicle speed and steering angle) for each point, In the attached pictures, the results show the path drawn based on the information entered. The red point shows the correct and appropriate path for this road and The black dots show the actual path of the vehicle. As show in **Figure 6.4**

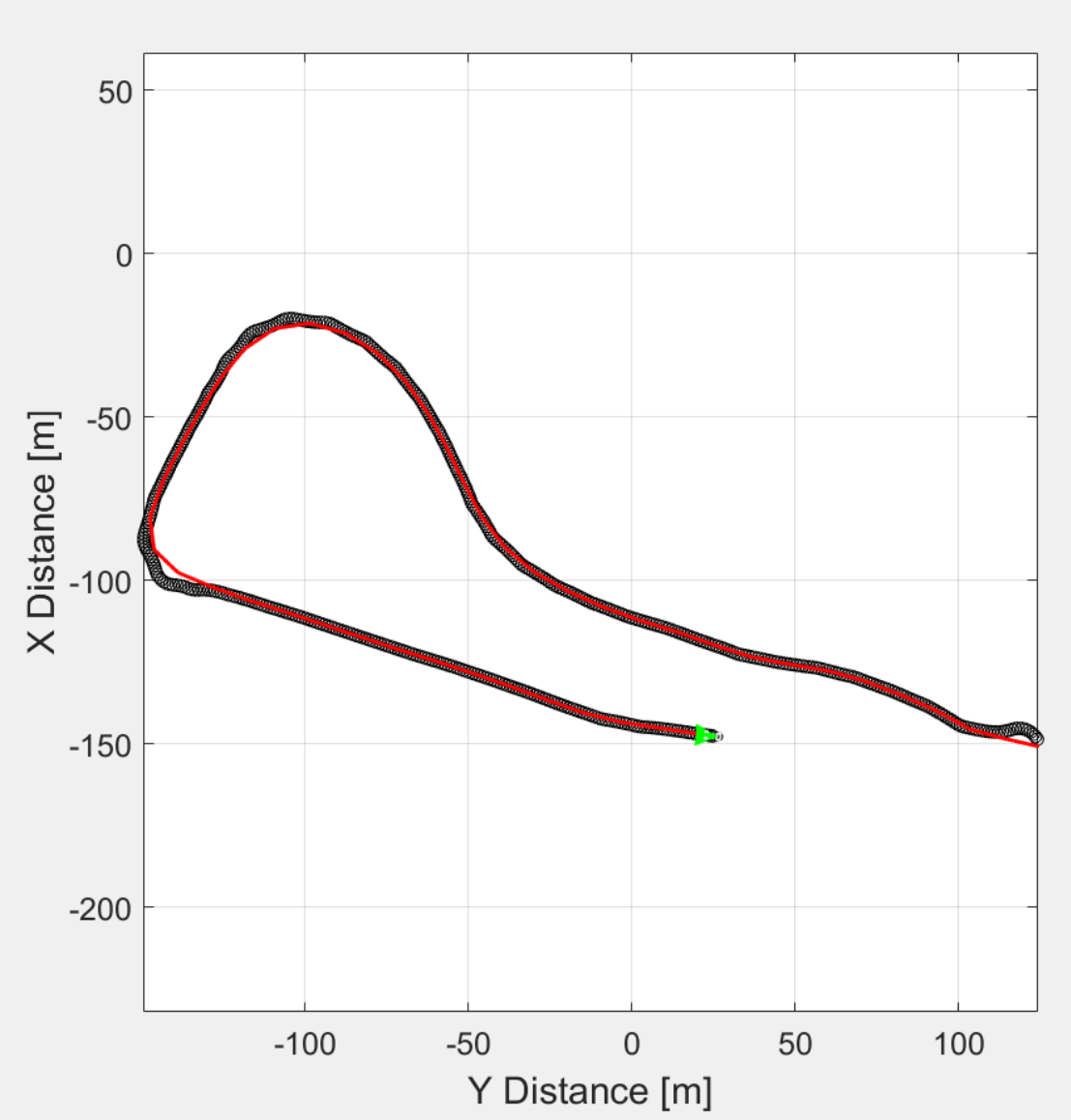


Figure 6.4 Result of simulation for the path

Sources of error and the percentage of error rises as a result of several main factors, and it turns out cannot obtain the ideal turn at high speeds and sharp turns, the model we use is 3 degree of freedom not 6 DOF like the real word and the some main input and constant of model like corner stiffness, center of gravity distance form ground …etc, this value take a general value from literature review , The **Figure 6.5** shown shows the big difference between the ideal turn and the turn resulting from the actual driving of the vehicle.

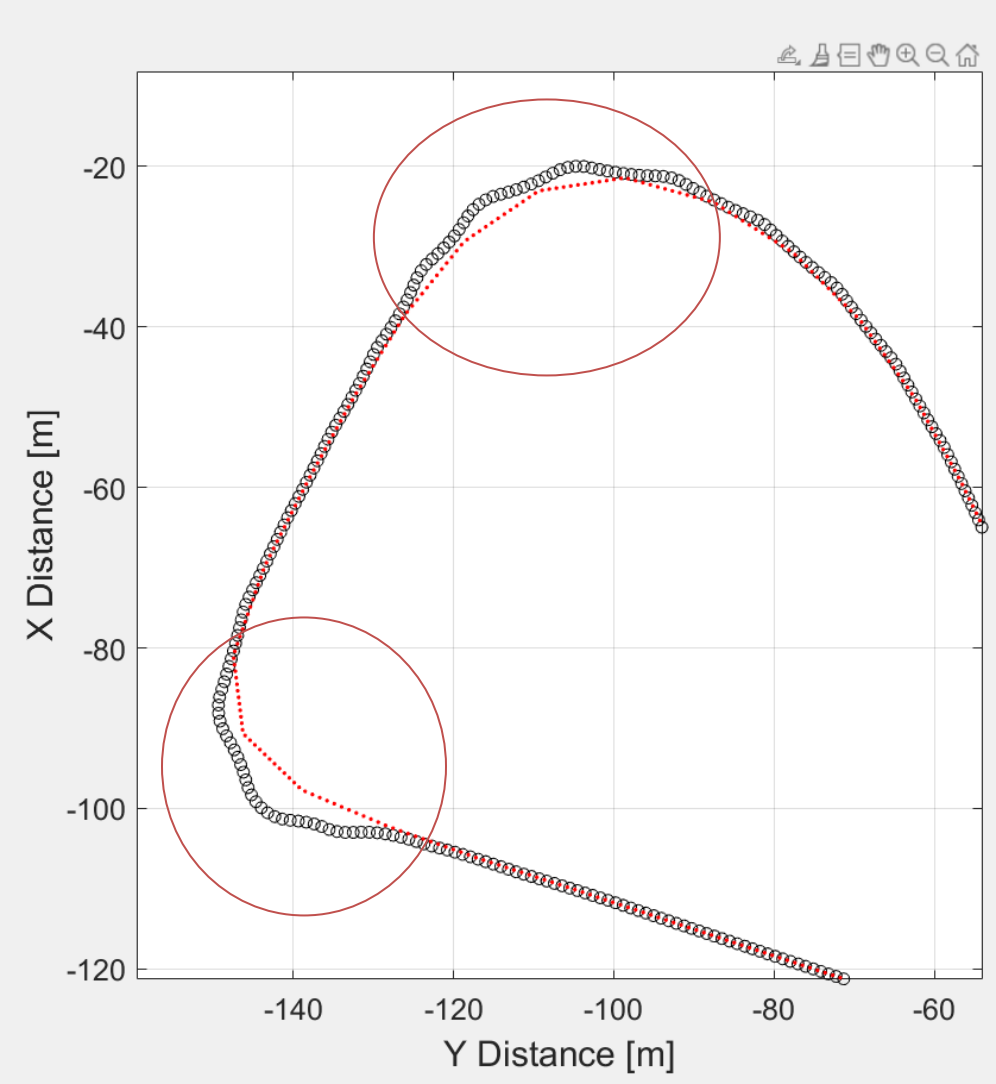


Figure 6.5 Error of simulation for the path

Through the scale and the use of the Vehicle Dynamics Toolbox 3DOF, results were obtained on verticals such as perfect speed, perfect steering angle, normal reaction front and rear (lift and right wheel), and Psi ,r (psi dot) which are shown in the **figure6.6 .**

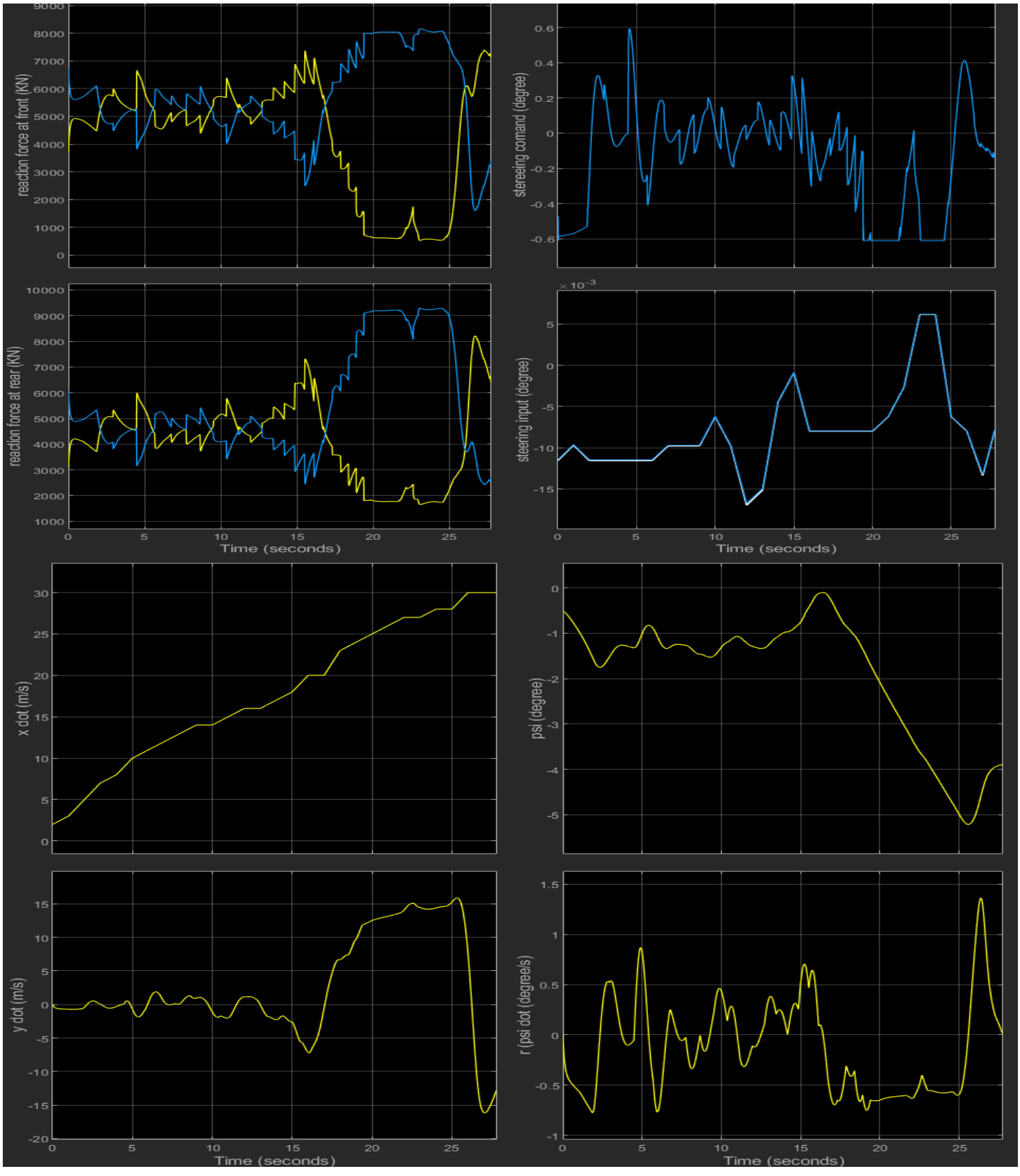


Figure 6.6 Result from 3DOFvehicle dynamic block

6.4 Recommendations

Due to several system and devices used in this project, there were a group of source of error coming out these system and devices, so, our recommendation come to solve these error and develop the system relative to these system and devices.

1. Related to GPS system

a. Using more accurate GPS sensor with a higher update rate too.

b. Using GPS sensor with more flexible antenna (longer antenna wire).

c. Linking the GPS system with weather condition station to receive the weather condition, which is a primary factor for computing the coefficient of friction.

d. Using a GPS sensor have a lower operation temperature, or a sensor reaching his operating temperature faster.

2. Related to GIS system

a. Prepare a GIS data for the rest of town, country.

b. Preparing more accurate GIS data by increasing the number of point coordinate per unit distance, and also using more accurate photogrammetry for both street side.

3. Related to the hardware component

a. Using faster micro controllers.
b. Exchange the Arduino micro controller with a micro processer, raspberry pi for example, which helps to dispense a personal laptop, and make the project closer to the final product.
c. Develop a new simple control module that can import the vehicle speed from the vehicle itself.
d. Try to use Seeed CAN Bus Shield V1.2 and learn how to select mood.

4. Related to Data extraction

Before choosing a vehicle, find out all the necessary information (**PID**) and make sure it is available.

For example, in our case, it is unable to find out the special PID in the vehicle's steering angle for the available vehicle

5. Related to calculations

a. Converting Model to Python code to facilitate real-time calculations and faster calculation.
b. Using more sophisticated mathematical model instead of our simple model, which will be more accurate.

6.5 The-State-of-the-Art of this technology

The rapid evolution in vehicle electronic systems including GPS, was consequent to the developing tools and information, according to this project came to develop a training module for safety and stability application, as it should , this system will be part of one or more future technology system, that serves the same object of this project, and as imagine that this project will integrate with some new technologies such as **Autonomous vehicles** By the combination of this system and another smart models include stability control systems, navigation system and vehicle control systems, it will achieve an Autonomous vehicle control, which means the vehicle drives without the need for a driver, this trend could be one of the most interested state of the art technology that is related to this project.

References

[1] Acea.auto. [Online]. Available: https://www.acea.auto/figure/world-motor-vehicle-production/.3-5-2022. [Accessed: 02-Aug-2022].

[2] GPS TECHNOLOGY OPTIMIZING CAR NAVIGATION Elvis N. Ngah Business Mathematics and Informatics Vrije. 2006.

[3] Y. Zein, M. Darwiche, and O. Mokhiamar, "GPS tracking system for autonomous vehicles," Alexandria engineering journal, 2018

[4] "The Global Positioning System: Signals, Measurements, and Performance Per K," Enge International Journal of Wireless Information Networks, vol. I, no. 2, 1994.

[5] A. El-Rabbany, Introduction to GPS: The global positioning system, 2nd ed. Norwood, MA: Artech House, 2006.

[6] S. Wise, GIS Basics. London, England: CRC Press, 2018.

[7] Van Sickle, Jan. Basic GIS coordinates. CRC press, 2017.

[8] R. Rajamani, Vehicle dynamics and control. Springer Science & Business Media, 2011.

[9] C. Miller and C. Valasek, A survey of remote automotive attack surfaces. Black Hat USA, 2014.

[10] M. di Natale, H. Zeng, P. di Giusto, and A. Ghosal, Understanding and using the controller area network communication protocol: Theory and practice, 2012th ed. New York, NY: Springer, 2012.

[11] CANedge: CAN Bus Data Logger - Simple-To-Use. Pro Specs. Interoperable. 2020.

[12] Y. Ding, "Three methods of vehicle lateral control: Pure pursuit, Stanley and MPC," Medium, 06-Mar-2020. [Online]. Available: https://dingyan89.medium.com/three-methods-of-vehicle-lateral-control-pure-pursuit-stanley-and-mpc-db8cc1d32081. [Accessed: 02-Aug-2022].

[13] Zein, Yassine; Darwiche, Mohamad; Mokhiamar, Ossama (2018). GPS tracking system for autonomous vehicles. Alexandria Engineering Journal, (), S1110016818301091–. doi: 10.1016/j.aej.2017.12.002

[14] Qais NaserEddin , Mohammad Badran , Tareq Dweik , Sondos Harfoush. (2016). Developing Training Module on GPS Applications in Automotive Safety and Stability. Palestine Polytechnic University (PPU), Birzeit University (BZU).

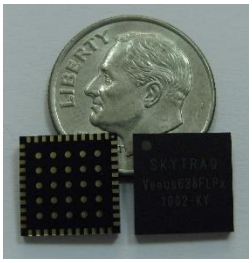
[15] “3DOF rigid vehicle body to calculate longitudinal, lateral, and yaw motion - Simulink,” Mathworks.com. [Online]. Available: https://www.mathworks.com/help/vdynblks/ref/vehiclebody3dof.html. [Accessed: 09-Aug-2022].

[16] “Road traffic injuries,” *Who.int*. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Accessed: 05-Sep-2022]

Appendix A

Venus638FLPx GPS Receiver

Data Sheet



10mmx 10mm

Venus638FLPx-L / Venus638FLPx-D

FEATURES

- 20Hz update rate
- -148dBm cold start sensitivity
- -165dBm tracking sensitivity
- 29 second cold start TTFF
- 3.5 second TTFF with AGPS
- 1 second hot start
- 2.5m accuracy
- Multipath detection and suppression
- Jamming detection and mitigation
- SBAS (WAAS / EGNOS) support
- 7-day extended ephemeris AGPS
- 67mW full power navigation
- Works directly with active or passive antenna
- Internal flash for optional 75K point data logging
- Supports external SPI flash memory data logging
- Complete receiver in 10mm x 10mm x 1.3mm size
- Contains LNA, SAW Filter, TCXO, RTC Xtal, LDO
- Pb-free RoHS compliant

Venus638FLPx is a high performance, low cost, single chip GPS receiver targeting mobile consumer and cellular handset applications. It offers very low power consumption, high sensitivity, and best in class signal acquisition and time-to-first-fix performance.

Venus638FLPx contains all the necessary components of a complete GPS receiver, includes 1.2dB cascaded system NF RF front-end, GPS baseband signal processor, 0.5ppm TCXO, 32.768kHz RTC crystal, RTC LDO regulator, and passive components. It requires very low external component count and takes up only 100mm^2 PCB footprint.

Dedicated massive-correlator signal parameter search engine within the baseband enables rapid search of all the available satellites and acquisition of very weak signal. An advanced track engine allows weak signal tracking and positioning in harsh environments such as urban canyons and under deep foliage.

The self-contained architecture keeps GPS processing off the host and allows integration into applications with very little resource.

Venus638FLPx is very easy to use, minimizes RF layout design issues and offers very fast time to market.

Receiver Type	L1 frequency GPS C/A code SBAS capable 65-channel architecture 8 million time-frequency searches per second
Accuracy	Position 2.5m CEP Velocity 0.1m/sec Timing 60ns
Open Sky TTFF	29 second cold start 3.5 second with AGPS 1 second hot start
Reacquisition	< 1s
Sensitivity	-165dBm tracking -148dBm cold start
Update Rate	1 / 2 / 4 / 5 / 8 / 10 / 20 Hz (default 1Hz)
Dynamics	4G
Operational Limits	Altitude < 18,000m[*1] , Velocity < 515m/s[*1]
Datum	Default WGS-84
Interface	UART LVTTL level
Baud Rate	4800 / 9600 / 38400 / 115200
Protocol	NMEA-0183 V3.01, GGA, GLL, GSA, GSV, RMC, VTG (default GGA, GSA, GSV, RMC, VTG) SkyTraq Binary
Main Supply Voltage	2.8V ~ 3.6V (Venus638FLPx-L) 2.8V ~ 3.6V, 1.08V ~ 1.32V (Venus638FLPx-D)
Backup Voltage	1.5V ~ 6V
Current Consumption	

	Enhanced Acquisition	Low Power Acquisition	Tracking
Venus638FLPx-L	68mA @ 3.3V	50mA @ 3.3V	29mA @ 3.3V
Venus638FLPx-D	18mA @ 3.3V 50mA @ 1.2V	18mA @ 3.3V 32mA @ 1.2V	18mA @ 3.3V 11mA @ 1.2V

Assuming 75% efficiency switch-mode 3.3V-to-1.2V regulator is used, then

	Enhanced Acquisition	Low Power Acquisition	Tracking

BLOCK DIAGRAM

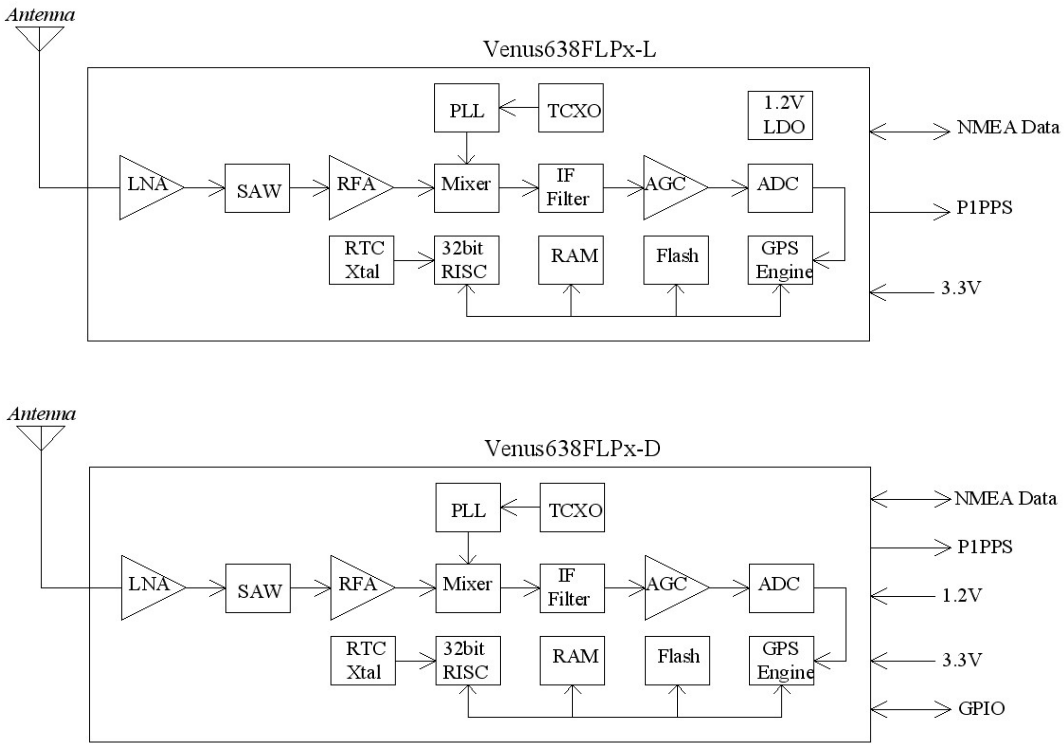


Figure-1 GPS Receiver based on Venus638FLPx

VENUS638FLPx PIN-OUT DIAGRAM

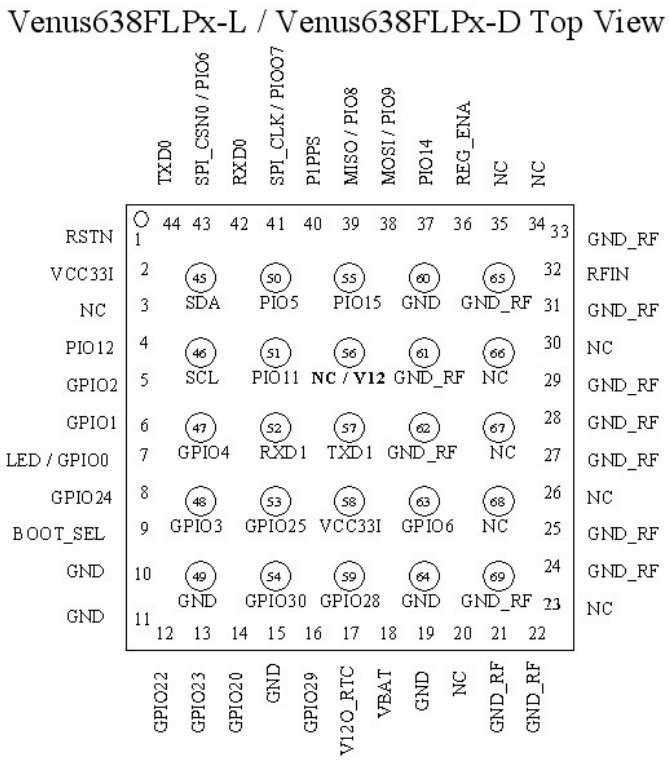


Figure-2b Venus638FLPx Pin-Out Diagram

VENUS638FLPx PIN DEFINITION

Pin Number	Signal Name	Type	Description
1	RSTN	Input	Active LOW reset input, 3.3V LVTTL
2	VCC33I	Power Input	Main voltage supply input, 2.8V ~ 3.6V
3	NC		Not connected, empty pin
4	PIO12	Bidir	General purpose I/O pin, 3.3V LVTTL
5	GPIO2	Bidir	General purpose I/O pin, 3.3V LVTTL
6	GPIO1	Bidir	General purpose I/O pin, 3.3V LVTTL
7	LED / GPIO0	Bidir	Navigation status indicator or General purpose I/O. 3.3V LVTTL

8	GPIO24	Bidir	General purpose I/O pin. 3.3V LVTTL Also serves as Search Engine Mode Selection upon power-up 1: low power acquisition mode 0: enhanced acquisition mode
9	BOOT_SEL	Bidir	Boot mode selection. Pull-high or pull-low using 10K resistor. Must not connect to VCC or GND directly. 1: execute from internal ROM 0: execute from internal Flash memory
10	GND	Power	System ground
11	GND	Power	System ground
12	GPIO22	Bidir	General purpose I/O pin, 3.3V LVTTL
13	GPIO23	Bidir	General purpose I/O pin, 3.3V LVTTL
14	GPIO20	Bidir	General purpose I/O pin, 3.3V LVTTL
15	GND	Power	System ground
16	GPIO29	Bidir	General purpose I/O pin, 3.3V LVTTL
17	V12O_RTC	Power Output	1.2V LDO output for RTC & backup memory. Normally unused.
18	VBAT	Power Input	Supply voltage for internal RTC and backup SRAM, 1.5V ~ 6V. VBAT should be powered by non-volatile supply voltage to have optimal performance. If VBAT is connected to VCC33I, powered off as VCC33I power is removed, then it'll cold start every time. For applications that do not care lesser performance cold starting every time, this pin can be connected to VCC33I.
19	GND	Power	System ground
20	NC		Not connected, empty pin
21	GND_RF	Power	RF section system ground
22	GND_RF	Power	RF section system ground
23	NC		Not connected, empty pin
24	GND_RF	Power	RF section system ground
25	GND_RF	Power	RF section system ground
26	NC		Not connected, empty pin
27	GND_RF	Power	RF section system ground
28	GND_RF	Power	RF section system ground
29	GND_RF	Power	RF section system ground
30	NC		Not connected, empty pin
31	GND_RF	Power	RF section system ground
32	RFIN	Input	GPS signal input, connect to GPS antenna.
33	GND_RF	Power	RF section system ground
34	NC		Not connected, empty pin
35	NC		Not connected, empty pin
36	REG_ENA	Input	Connect to pin-2 VCC33I
37	PIO14	Bidir	General purpose I/O pin, 3.3V LVTTL
38	MOSI / PIO9	Bidir	SPI data output or general purpose I/O pin, 3.3V LVTTL
39	MISO / PIO8	Bidir	SPI data input or general purpose I/O pin, 3.3V LVTTL
40	P1PPS	Output	1 pulse per second output. Active after position fix; goes HIGH for about 4msec, 3.3V LVTTL
41	SPI_CLK / PIOO7	Output	SPI clock or general purpose output pin, 3.3V LVTTL
42	RXD0	Input	Received input of the asynchronous UART port. Used to input binary command to the GPS receiver. 3.3V LVTTL
43	SPI_CSN / PIO6	Bidir	SPI chip select output or general purpose I/O pin, 3.3V LVTTL
44	TXD0	Output	Transmit output of the asynchronous UART port. Used to output standard NMEA-0183 sentence or response to input binary command. 3.3V LVTTL

45	SDA	Bidir	I2C data, 3.3V I/O
46	SCL	Bidir	I2C clock, 3.3V I/O
47	GPIO4	Bidir	General purpose I/O pin, 3.3V LVTTL
48	GPIO3	Bidir	General purpose I/O pin, 3.3V LVTTL
49	GND		System ground
50	PIO5	Output	General purpose output pin, 3.3V LVTTL
51	PIO11	Bidir	General purpose I/O pin, 3.3V LVTTL
52	RXD1	Input	Received input of the asynchronous UART port. 3.3V LVTTL
53	GPIO25	Bidir	General purpose I/O pin, 3.3V LVTTL
54	GPIO30	Bidir	General purpose I/O pin, 3.3V LVTTL
55	PIO15	Bidir	General purpose I/O pin, 3.3V LVTTL
56	NC / V12		NC pin for Venus638FLPx-L 1.2V supply input pin for Venus638FLPx-D
57	TXD1	Output	Transmit output of the asynchronous UART port. 3.3V LVTTL
58	VCC33I	Power Input	Main voltage supply input, 2.8V ~ 3.6V
59	GPIO28	Bidir	General purpose I/O pin, 3.3V LVTTL
60	GND	Power	System ground
61	GND_RF	Power	RF section system ground
62	GND_RF	Power	RF section system ground
63	GPIO6	Bidir	General purpose I/O pin, 3.3V LVTTL
64	GND	Power	System ground
65	GND_RF	Power	RF section system ground
66,67,68	NC		
69	GND_RF	Power	RF section system ground

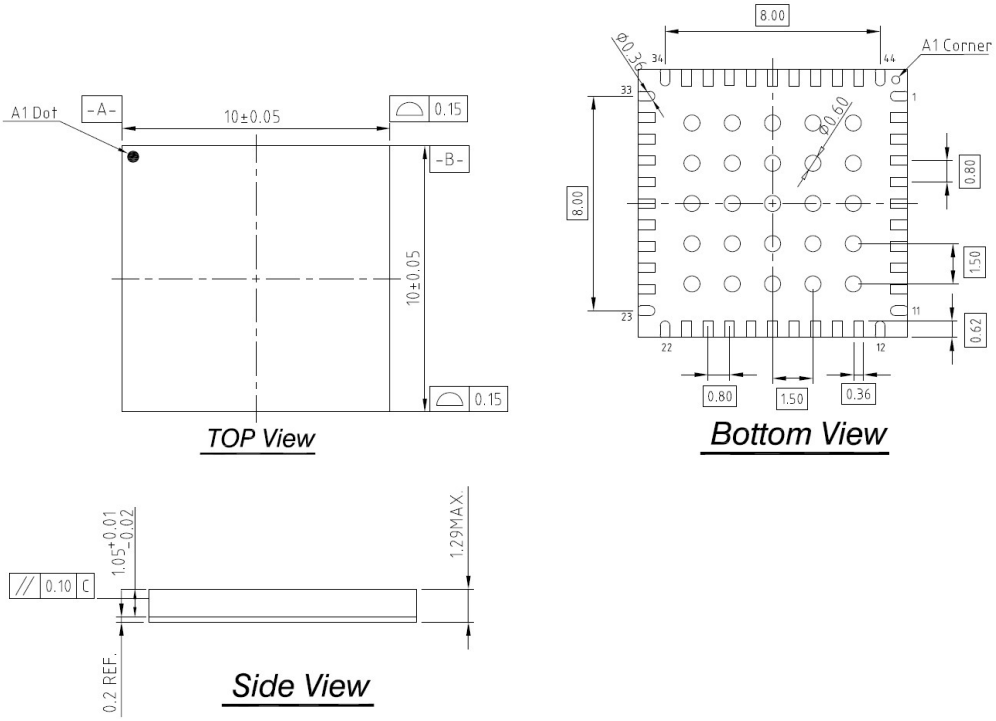
When using Venus638FLPx-L to replace Venus634FLPx, pin-45 ~ pin-69 can all be left unconnected. When using Venus638FLPx-D, 1.2V need to be supplied at pin-56 The NC pins are to be left unconnected.

DC CHARACTERISTICS OF DIGITAL INTERFACE

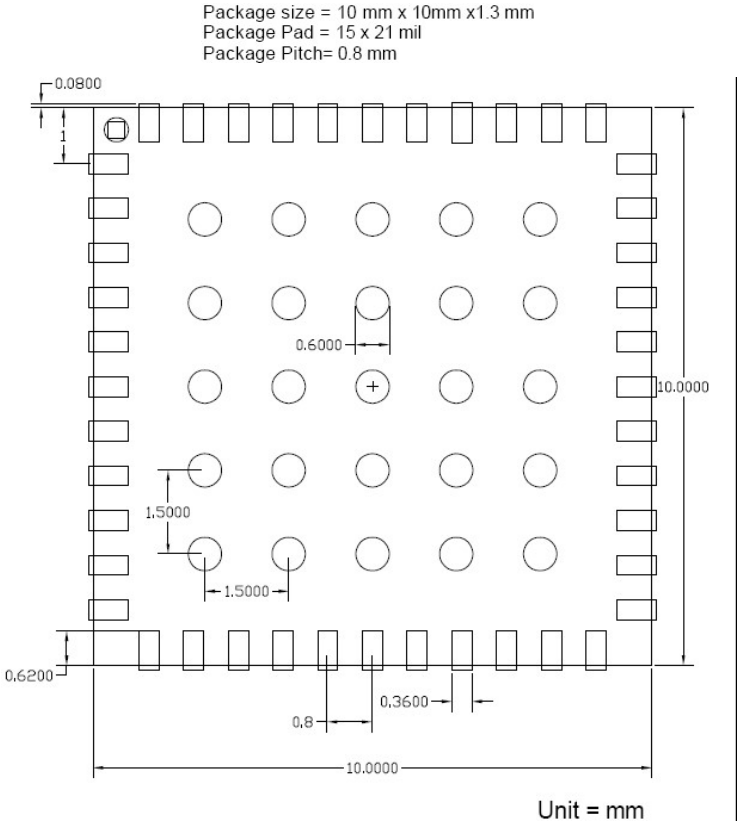
Below is when VCC3I is at nominally 3.3V

Parameter	**Min.**	**Typ.**	**Max.**	**Units**
Input Low Voltage			0.8	Volt
Input High Voltage	2.0			Volt
Output Low Voltage, Iol = 2 ~ 16mA			0.4	Volt
Output High Voltage, Ioh = 2 ~ 16mA	2.9			Volt

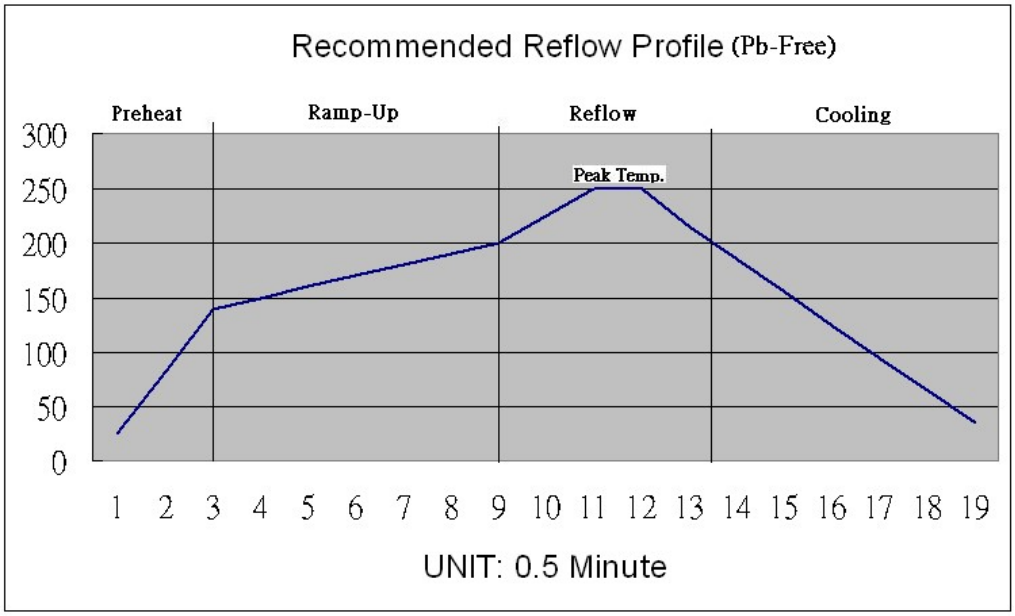
MECHANICAL DIMENSION



RECOMMENDED PCB FOOTPRINT



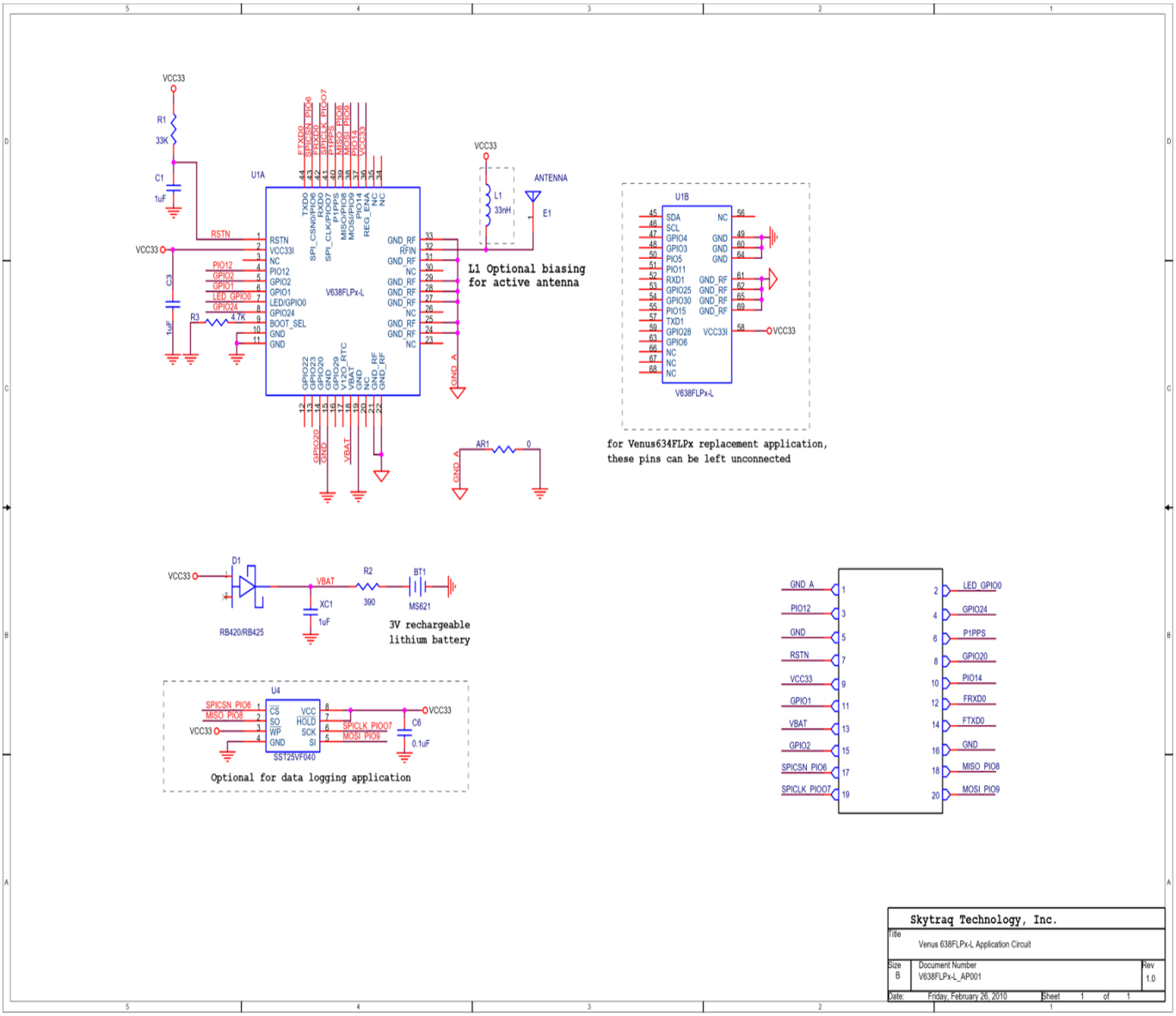
RECOMMENDED REFLOW PROFILE



Temperature (℃)	25	82.5	140	150	160	170	180	190	200	225	250	250	215	185	155	125	95	65	35
Time(minute)	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9

Profile Description	SnPb Eutectic Process	Lead Free Process
Preheat		
Maximum Temperature	100+/-10 ℃	140+/-10 ℃
Time(△T)	40~60s	50~70s
Ramp-Up		
Ramp-Up Rate	1 ℃/s Max.	1 ℃/s Max.
Time(△T)	120~150s	160~200s
Reflow		
Maximum Temperature	Peak Temp.	Peak Temp.
Minimum Temperature	180+/-5℃	200+/-10℃
Peak Temperature	220+/-2℃	250+/-2℃

Time(△T) during Peak Temp.+/-2℃	10~30s	20~40s
Reflow Time(△T)	120~150s	120~150s
Cooling		
Cooling Rate	1.5 ℃/s Max	1.5 ℃/s Max
Time(△T)	60~120s	150~180s



APPLICATION CIRCUIT INTERFACE SIGNALS

GND_A:	RF ground
LED:	Signal to indicate GPS position status, 3.3V LVTTL. Active low for no-fix, toggle every second after position fix.
PSE_SEL:	Search engine mode selection, sampled only at end of power-on reset cycle 1: Low power acquisition mode 0: Enhanced acquisition mode
GND:	Digital ground
P1PPS:	1 pulse per second time-mark (3.3V LVTTL)
RSTN:	Active low reset input
VCC33:	3.3V power input
FRXD0:	UART input (3.3V LVTTL)
FTXD0:	UART output (3.3V LVTTL)
VBAT:	Battery-backed RTC and SRAM supply input, 1.5V ~ 6V, must not be unconnected.

APPLICATION INFORMATION

1. For fast-rising power supply, a simple series R/C reset delay to pin-1, RSTN, as indicated in the application circuit is suitable. For system having slow-rising power supply, a reset IC providing 2~5ms reset duration may be necessary.

2. The RF input of Venus638FLPx is already matched to 50-ohm. Passive antenna matched to 50-ohm can be directly applied.

3. For using Venus638FLPx with active antenna, one with gain in range of 10~30dB and noise figure < 2dB can be used. Power to the active antenna needs to be applied externally.

4. Pin-18 VBAT supplies backup power to the real-time clock and backup SRAM for fast startup. For portable applications where there is battery with voltage in range of 1.5V ~ 6.0V as the main source, the VBAT pin can be directly connected to it. If VBAT is connected to main power as pin-2, no supply voltage as Venus638FLPx is powered off, then it'll cold start every time and GPS performance will not be optimal.

5. Like BGA device, the Venus638FLPx is moisture sensitive. It needs to be handled with care to void damage from moisture absorption and SMT re-flow. The device should be baked for 24 hours at 125-degC before mounting for SMT re-flow if it has been removed from the protective seal for more than 48[*1]hours.

6. The supported SPI Flash memory verified for data logging application are:

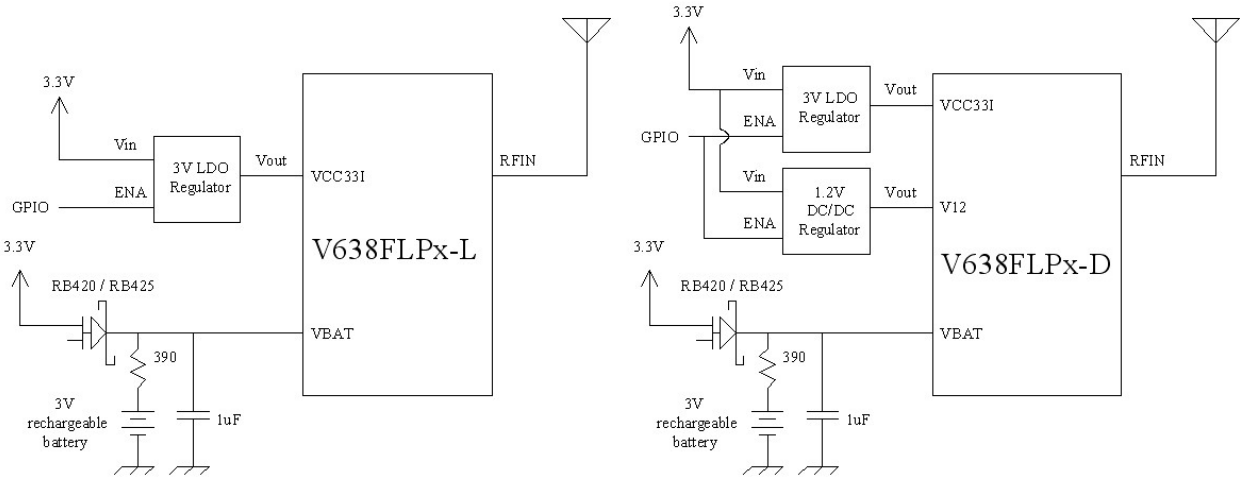
Manufacturer	Device ID	Size
EON	EN25F040	4Mbit
EON	EN25F080	8Mbit MXIC
MX25L400	4Mbit	
MXIC	MX25L800	8Mbit
MXIC	MX25L1605	16Mbit
MXIC	MX25L3205	32Mbit
MXIC	MX25L6405	64Mbit
WINBOND	W25X40	4Mbit
WINBOND	W25X80	8Mbit
WINBOND	W25X16	16Mbit
WINBOND	W25X32	32Mbit
WINBOND	W25X64	64Mbit
SST	SST25LF040	4Mbit
SST	SST25LF080	8Mbit
SST	SST25VF016	16Mbit
SST	SST 25VF032	32Mbit

7. The P1PPS pin must not be pulled-high during power on reset, or it'll enter into debug mode and freeze.

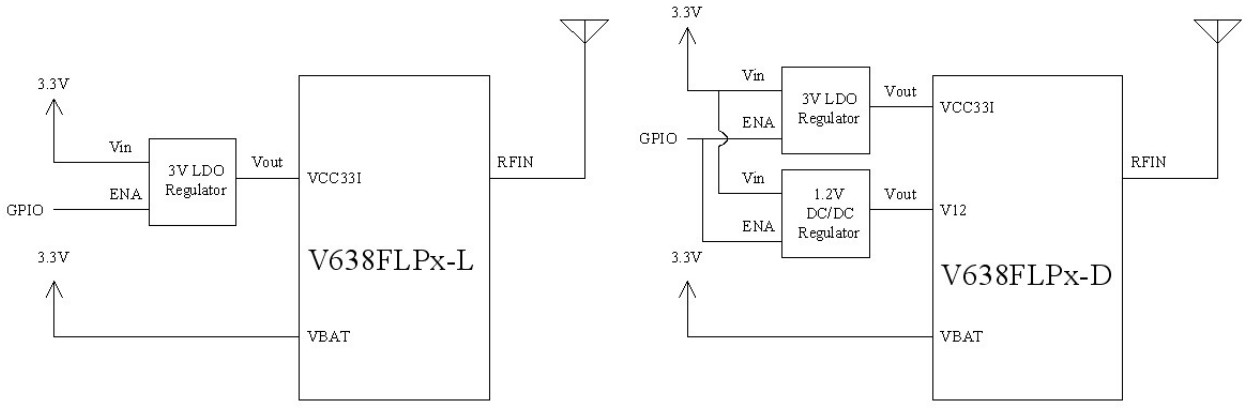
*1: Actual will be longer, moisture sensitivity level still undergoing verification.

SLEEP MODE

For application requiring sleep mode, it can be implemented using regulator with enable control as below figure shows. To put Venus638FLPx to sleep, the power to Venus638FLPx is cut off by disabling the regulator via host processor GPIO pin. In sleep mode, VBAT consume less than 10uA. Fast start up operation is provided by keeping supply voltage to VBAT constant, retaining the internal data and keep RTC running while Venus638FLPx is put to sleep or when supply 3.3V power is removed.



For applications needing sleep mode but cannot have extra cost of adding a rechargeable backup supply battery, it can be implemented as below figure shows. It will provide fast start up when Venus638FLPx is put to sleep and awakened, but will cold start every time when the 3.3V supply voltage is removed and re-applied again.



When using sleep mode, add 10K series resistor on pin-42 RXD0 and pin-44 TXD0.

NMEA MESSAGES

The full descriptions of supported NMEA messages are provided at the following paragraphs.

GGA - Global Positioning System Fix Data

Time, position and fix related data for a GPS receiver.

Structure:

$GPGGA,hhmmss.sss,ddmm.mmmm,a,dddmm.mmmm,a,x,xx,x.x,x.x,M,,,,xxxx*hh<CR><LF>

1 2 3 4 5 6 7 8

9 10 11

Example:

$GPGGA,111636.932,2447.0949,N,12100.5223,E,1,11,0.8,118.2,M,,,,0000*02<CR><LF>

Field	Name	Example	Description
1	UTC Time	111636.932	UTC of position in hhmmss.sss format, (000000.000 ~ 235959.999)
2	Latitude	2447.0949	Latitude in ddmm.mmmm format Leading zeros transmitted
3	N/S Indicator	N	Latitude hemisphere indicator, 'N' = North, 'S' = South
4	Longitude	12100.5223	Longitude in dddmm.mmmm format Leading zeros transmitted
5	E/W Indicator	E	Longitude hemisphere indicator, 'E' = East, 'W' = West
6	GPS quality indicator	1	GPS quality indicator 0: position fix unavailable 1: valid position fix, SPS mode 2: valid position fix, differential GPS mode 3: GPS PPS Mode, fix valid 4: Real Time Kinematic. System used in RTK mode with fixed integers 5: Float RTK. Satellite system used in RTK mode. Floating integers 6: Estimated (dead reckoning) Mode 7: Manual Input Mode 8: Simulator Mode
7	Satellites Used	11	Number of satellites in use, (00 ~ 12)
8	HDOP	0.8	Horizontal dilution of precision, (00.0 ~ 99.9)
9	Altitude	108.2	mean sea level (geoid), (-9999.9 ~ 17999.9)
10	DGPS Station ID	0000	Differential reference station ID, 0000 ~ 1023 NULL when DGPS not used

11	Checksum	02	

GLL – Latitude/Longitude

Latitude and longitude of current position, time, and status.

Structure:

$GPGLL,ddmm.mmmm,a,dddmm.mmmm,a,hhmmss.sss,A,a*hh<CR><LF>

1 2 3 4 5 6 7 8

Example:

$GPGLL,2447.0944,N,12100.5213,E,112609.932,A,A*57<CR><LF>

Field	Name	Example	Description
1	Latitude	2447.0944	Latitude in ddmm.mmmm format Leading zeros transmitted
2	N/S Indicator	N	Latitude hemisphere indicator 'N' = North 'S' = South
3	Longitude	12100.5213	Longitude in dddmm.mmmm format Leading zeros transmitted
4	E/W Indicator	E	Longitude hemisphere indicator 'E' = East 'W' = West
5	UTC Time	112609.932	UTC time in hhmmss.sss format (000000.000 ~ 235959.999)
6	Status	A	Status, 'A' = Data valid, 'V' = Data not valid
7	Mode Indicator	A	Mode indicator 'N' = Data not valid 'A' = Autonomous mode 'D' = Differential mode 'E' = Estimated (dead reckoning) mode 'M' = Manual input mode 'S' = Simulator mode
8	Checksum	57	

GSV – GNSS Satellites in View

Number of satellites (SV) in view, satellite ID numbers, elevation, azimuth, and SNR value. Four satellites maximum per transmission.

Structure:

$GPGSV,x,x,xx,xx,xx,xxx,xx,…,xx,xx,xxx,xx *hh<CR><LF>

1 2 3 4 5 6 7 4 5 6

7 8

Example:

$GPGSV,3,1,12,05,54,069,45,12,44,061,44,21,07,184,46,22,78,289,47*72<CR><LF>

$GPGSV,3,2,12,30,65,118,45,09,12,047,37,18,62,157,47,06,08,144,45*7C<CR><LF>

$GPGSV,3,3,12,14,39,330,42,01,06,299,38,31,30,256,44,32,36,320,47*7B<CR><LF>

Field	Name	Example	Description
1	Number of message	3	Total number of GSV messages to be transmitted (1-3)
2	Sequence number	1	Sequence number of current GSV message
3	Satellites in view	12	Total number of satellites in view (00 ~ 12)
4	Satellite ID	05	Satellite ID number, GPS: 01 ~ 32, SBAS: 33 ~ 64 (33 = PRN120)
5	Elevation	54	Satellite elevation in degrees, (00 ~ 90)
6	Azimuth	069	Satellite azimuth angle in degrees, (000 ~ 359)
7	SNR	45	C/No in dB (00 ~ 99) Null when not tracking
8	Checksum	72	

GSA – GNSS DOP and Active Satellites

GPS receiver operating mode, satellites used in the navigation solution reported by the GGA or GNS sentence and DOP values.

Structure:

$GPGSA,A,x,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,x.x,x.x,x.x*hh<CR><LF>

1 2 3 3 3 3 3 3 3 3 3 3 3 3

4 5 6 7

Example:

$GPGSA,A,3,05,12,21,22,30,09,18,06,14,01,31,,1.2,0.8,0.9*36<CR><LF>

Field	Name	Example	Description
1	Mode	A	Mode 'M' = Manual, forced to operate in 2D or 3D mode 'A' = Automatic, allowed to automatically switch 2D/3D
2	Mode	3	Fix type 1 = Fix not available 2 = 2D 3 = 3D
3	Satellite used 1~12	05,12,21,22,30,09,18,06,14,01,31,,	Satellite ID number, 01 to 32, of satellite used in solution, up to 12 transmitted
4	PDOP	1.2	Position dilution of precision (00.0 to 99.9)
5	HDOP	0.8	Horizontal dilution of precision (00.0 to 99.9)
6	VDOP	0.9	Vertical dilution of precision (00.0 to 99.9)
7	Checksum	36	

RMC – Recommended Minimum Specific GNSS Data

Time, date, position, course and speed data provided by a GNSS navigation receiver.

Structure:

$GPRMC,hhmmss.sss,A,dddmm.mmmm,a,dddmm.mmmm,a,x.x,x.x,ddmmyy,,,a*hh<CR><LF>

1 2 3 4 5 6 7 8 9 10 11

Example:

$GPRMC,111636.932,A,2447.0949,N,12100.5223,E,000.0,000.0,030407,,,A*61<CR><LF>

Field	Name	Example	Description
1	UTC time	0111636.932	UTC time in hhmmss.sss format (000000.00 ~ 235959.999)
2	Status	A	Status 'V' = Navigation receiver warning 'A' = Data Valid
3	Latitude	2447.0949	Latitude in dddmm.mmmm format Leading zeros transmitted
4	N/S indicator	N	Latitude hemisphere indicator 'N' = North 'S' = South
5	Longitude	12100.5223	Longitude in dddmm.mmmm format Leading zeros transmitted
6	E/W Indicator	E	Longitude hemisphere indicator 'E' = East 'W' = West
7	Speed over ground	000.0	Speed over ground in knots (000.0 ~ 999.9)
8	Course over ground	000.0	Course over ground in degrees (000.0 ~ 359.9)
9	UTC Date	030407	UTC date of position fix, ddmmyy format

10	Mode indicator	A	Mode indicator 'N' = Data not valid 'A' = Autonomous mode 'D' = Differential mode 'E' = Estimated (dead reckoning) mode 'M' = Manual input mode 'S' = Simulator mode
11	checksum	61	

VTG – Course Over Ground and Ground Speed

The Actual course and speed relative to the ground.

Structure:

GPVTG,x.x,T,,M,x.x,N,x.x,K,a*hh<CR><LF>

1 2 3 4 5

Example:

$GPVTG, 000.0,T,,M,000.0,N,0000.0,K,A*3D<CR><LF>

Field	Name	Example	Description
1	Course	000.0	True course over ground in degrees (000.0 ~ 359.9)
2	Speed	000.0	Speed over ground in knots (000.0 ~ 999.9)
3	Speed	0000.0	Speed over ground in kilometers per hour (0000.0 ~ 1800.0)
4	Mode	A	Mode indicator 'N' = not valid 'A' = Autonomous mode 'D' = Differential mode 'E' = Estimated (dead reckoning) mode 'M' = Manual input mode 'S' = Simulator mode
5	Checksum	3D	

ORDERING INFORMATION

Part Number	Description
Venus638FLPx-L	Flash version GPS receiver (internal 1.2V LDO version)
Venus638FLPx-D	Flash version GPS receiver (external 1.2V version)

SkyTraq Technology, Inc.
4F, No.26, Minsiang Street, Hsinchu, Taiwan, 300
Phone: +886 3 5678650
Fax: +886 3 5678680
Email: info@skytraq.com.tw

Appendix B

multi-turn potentiometer Data sheet

Features

- Multiturn / Wirewound / Industrial
- Metal shaft and bushings
- Bushing mount
- Available with shaft (6mm dia.)

Trimmer™

WXD3540 –Trimmer™ Precision Potentiometer

Electrical Characteristics

Standard Resistance Range
100 to 100K ohms
(see standard resistance table)
Resistance Tolerance
±5% std.
Independent Linearity
±0.25%
Effective Electrical Angle
3600° -10° min.
Absolute Minimum Resistance
0.2% or 2 ohms max.
(whichever is greater)
Noise
100 ohms ENR max.
Insulation Resistance
500 vdc
1,000 megohms min.
Dielectric Withstanding Voitage
101.3 kPa 900 vac

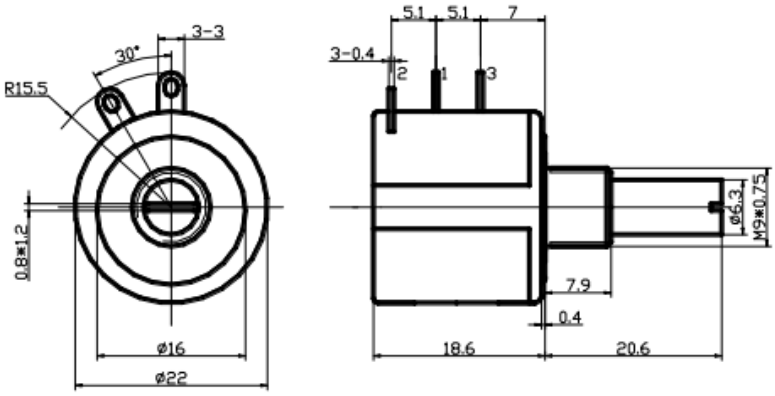
Environmental Characteristics

Power Rating (500 volts max.)
2 watt (70℃) , 0 watt (125℃)
Temperature Range
-55℃ to +125℃
Temperature Coefficient
±100ppm/℃
Vibration
98 m/s²
(1% Δ TR;1% Δ VR)
Shock
490 m/s²
(1% Δ TR; 1% Δ VR)
Load Life
1,000 hours 2 watt @ 70℃
(2% Δ TR)
Rotational Life
10000 cycles
(3% Δ TR)

Physical Characteristics

Torque
36 mN • m max.
Stop Strength
300 mN • m min.
Mechanical angle
3600° -10° min.
Standard Packaging
10 or 25 pcs. per box

WXD3540 Dimensions

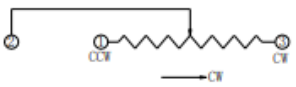


How to order

WXD3540—101

Model

Resistance Code



standard Resistance Table

Resistance (Ohms)	Resistance Code
100	101
200	201
500	501
1,000	102
2,000	202
5,000	502
10,000	103
20,000	203
50,000	503
100,000	104

Special resistance available.

Detail Specification: Q/RY531-2004

Appendix C

- **Working code for converting WGS 84 Format into UTM (python)**

```
import serial

import utm

import math

import keyboard


def latlon_to_xyz(lat,lon):

    """

    Convert angluar to cartesian coordiantes


    latitude is the 90deg - zenith angle in range [-90;90]

    lonitude is the azimuthal angle in range [-180;180]

    """

    r = 6371 # https://en.wikipedia.org/wiki/Earth_radius

    theta = math.pi/2 - math.radians(lat)

    phi = math.radians(lon)

    x = r * math.sin(theta) * math.cos(phi) # bronstein (3.381a)

    y = r * math.sin(theta) * math.sin(phi)

    z = r * math.cos(theta)

    return [x,y,z]

serialPort = serial.Serial(port = "COM8", baudrate=9600)

serialString = ""

lat=""

long=""
```

```
with open('awfa.txt','w') as file:
   with open('rami.txt','w') as rami:
      while(1):
         if keyboard.is_pressed("q"):
            break
         # Wait until there is data waiting in the serial buffer
         if(serialPort.in_waiting > 0):

            # Read data out of the buffer until a carraige return / new line is found
            serialString = serialPort.readline()
            lat=serialString[5:16]
            long=serialString[24:35]
            lat2=lat.decode('Ascii')
            long2=long.decode('Ascii')
            flat=float(lat2)
            flong=float(long2)
            u = latlon_to_xyz(flat,flong)
            print(serialString.decode('Ascii'))
            print(u)
            file.write(str(u)+"\n")
            rami.write(serialString.decode('Ascii'))
      # Print the contents of the serial data
 # print(serialString.decode('Ascii'))



    # serialPort.write(b"Thank you for sending data
```

Appendix D

- **The code used to extract data from the electronic control unit ECU (Speed and RPM) - Python**

```
import obd
import time
import serial
import keyboard
def replace_none_with_prev(values):
  prev_value = None
  for i, value in enumerate(values):
    if value is None:
      values[i] = prev_value
    else:
      prev_value = value
  return values
connection = obd.OBD('\\.\\COM5',fast=False)
serialPort = serial.Serial(port = "COM7", baudrate=9600)
sp=[]
rp=[]
serialString = ""
i=0
with open('awfa.txt','w') as file:
     with open('salem.txt','w') as file2:
        cmd  = obd.commands['SPEED']
```

```
cmd2  = obd.commands['RPM']
while (1):
        if keyboard.is_pressed('q'):
            break
        startgps=time.time()
        if(serialPort.in_waiting > 0):
            serialString = serialPort.readline()
            # endgps=time.time()
            print(serialString.decode('Ascii'),"  time gps",(startgps))
            file2.write(serialString.decode('Ascii')+"\t "+str(startgps)+" ")
        start = time.time()
        response = connection.query(cmd)
      sp.insert(i,response)
      replace_none_with_prev(sp)
      response2 = connection.query(cmd2)
      rp.insert(i,response2)
      replace_none_with_prev(rp)
      print("\t",str(response.value),"\t",response2.value,)
      # +"\t",response2.value
      # send the command, and parse the response
      i=i+1
      file.write(str(response.value)+"\t"+str(response2.value)+"\t" "time :"
+str(start)+"\n")
  # file.write("time"+str(start))
```

```
        # end = time.time(

        # print(response.value," time obd",(end-start))

# print("the elapsed time of gps sensor and obd command response :", (end-start) )
```

Appendix E

- **Setup model (MATLAB code)**

Add images to the path

```
addpath(genpath('Images'));
```

Load scene data files

load data from Driving Scenario Designer

```
load('gis trak.mat'); % GIS file m.file
refPose = data.ActorSpecifications.Waypoints;
```

Define reference points

```
xRef = refPose(:,1);
yRef = -refPose(:,2);
```

Define vehicle parameters

```
X_o = xRef(1); % initial vehicle position in x direction
Y_o = yRef(1); % initial vehicle position in y direction
```

Calculating reference pose vectors

Based on how far the vehicle travels, the pose is generated using 1-D lookup tables.

```
% calculate distance vector
distancematrix = squareform(pdist(refPose));
distancesteps = zeros(length(refPose)-1,1);
for i = 2:length(refPose)
    distancesteps(i-1,1) = distancematrix(i,i-1);
end
totalDistance = sum(distancesteps); % Total distance travelled
distbp = cumsum([0; distancesteps]); % Distance for each waypoint
gradbp = linspace(0,totalDistance,50); % Linearize distance


% linearize X and Y vectors based on distance
xRef2 = interp1(distbp,xRef,gradbp);
yRef2 = interp1(distbp,yRef,gradbp);
yRef2s = smooth(gradbp,yRef2); % smooth waypoints
xRef2s = smooth(gradbp,xRef2); % smooth waypoints
```

```
plot(gradbp,xRef2s)
xlabel('distance')
ylabel('x')
plot(gradbp,yRef2s)
xlabel('distance')
ylabel('y')
```

Calculate theta vector

theta = orientation angle of the path at reference points

```
thetaRef = zeros(length(gradbp),1);
for i = 2:length(gradbp)
    thetaRef(i,1) = atan2d((yRef2(i)-yRef2(i-1)),(xRef2(i)-xRef2(i-1)));
end
thetaRefs = smooth(gradbp,thetaRef); % smooth of theta
psi_o = thetaRefs(1)*(pi/180); % initial yaw angle
plot(gradbp,thetaRefs)
xlabel('distance')
ylabel('theta')
```

Create direction vector

```
direction = ones(length(gradbp),1);
```

Calculate curvature vector

```
curvature = getCurvature(xRef2,yRef2);
plot(gradbp,curvature)
xlabel('distance')
ylabel('curvature')
```

Curvature Function

```
function curvature = getCurvature(xRef,yRef)
% Calculate gradient by the gradient of the X and Y vectors
DX = gradient(xRef);
D2X = gradient(DX);
DY = gradient(yRef);
D2Y = gradient(DY);
curvature = (DX.*D2Y - DY.*D2X) ./(DX.^2+DY.^2).^(3/2);
end
```

Appendix F

- **Canbus shield code (Arduino C)**
  - **Code#1**

```
// demo: CAN-BUS Shield, receive data with check mode
// send data coming to fast, such as less than 10ms, you can use this way
#include <SPI.h>
#include "mcp_can.h"
#define CAN_2515
// #define CAN_2518FD
// Set SPI CS Pin according to your hardware
#if defined(SEEED_WIO_TERMINAL) && defined(CAN_2518FD)
// For Wio Terminal w/ MCP2518FD RPi Hat：
// Channel 0 SPI_CS Pin: BCM 8
// Channel 1 SPI_CS Pin: BCM 7
// Interupt Pin: BCM25
const int SPI_CS_PIN  = BCM8;
const int CAN_INT_PIN = BCM25;
#else

// For Arduino MCP2515 Hat:
// the cs pin of the version after v1.1 is default to D9
// v0.9b and v1.0 is default D10
const int SPI_CS_PIN = 10;
const int CAN_INT_PIN = 2;
#endif

/*#ifdef CAN_2518FD
#include "mcp2518fd_can.h"
mcp2518fd CAN(SPI_CS_PIN); // Set CS pin
#endif
*/
#ifdef CAN_2515
#include "mcp2515_can.h"
mcp2515_can CAN(10); // Set CS pin

#endif
```

```
void setup() {
  SERIAL_PORT_MONITOR.begin(115200);
  while (CAN_OK != CAN.begin(CAN_500KBPS)) {          // init can bus : baudrate = 500k
    SERIAL_PORT_MONITOR.println("CAN init fail, retry...");
    delay(100);
  }
  SERIAL_PORT_MONITOR.println("CAN init ok!");
}
void loop() {
  unsigned char len = 0;
  unsigned char buf[8];
  if (CAN_MSGAVAIL == CAN.checkReceive()) {        // check if data coming
    CAN.readMsgBuf(&len, buf);   // read data,  len: data length, buf: data buf

    unsigned long canId = CAN.getCanId();

    SERIAL_PORT_MONITOR.println("-----------------------------");
    SERIAL_PORT_MONITOR.print("Get data from ID: 0x");
    SERIAL_PORT_MONITOR.println(canId, HEX);


    for (int i = 0; i < len; i++) { // print the data
      SERIAL_PORT_MONITOR.print(buf[i], HEX);
      SERIAL_PORT_MONITOR.print("\t");
    }
    SERIAL_PORT_MONITOR.println();
  }
}

/*********************************************************************
*******************************
  END FILE
*********************************************************************
*******************************/
*********************************************************************
```

Code#2

```
#include <Arduino.h>
#include <mcp_can.h>
```

```
#include <mcp_can_dfs.h>

#define CANint 2
#define LED2 8
#define LED3 7
#define MCP_STDEXT 0
unsigned char len = 0;
unsigned char buf[8];
unsigned long ID = 0;
unsigned long line = 0;

MCP_CAN CAN0(10); // Set CS to pin

unsigned long time;
void setup() {
 Serial.begin(115200);

 while (!Serial) {
  Serial.print("I will wait here forever...");
    delay(1000);
 };

 pinMode(23, OUTPUT);
 digitalWrite(23, HIGH);

 pinMode(LED2, OUTPUT);
 pinMode(LED3, OUTPUT);
 pinMode(CANint, INPUT);
 digitalWrite(LED2, LOW);

 Serial.println("CAN init:");

 if (CAN0.begin(CAN_500KBPS) == CAN_OK) {
  Serial.println("Can Init Success");
 } else {
  Serial.println("Can Init Failed");
  while (1) {
   Serial.print("I will wait here forever...");
```

```
    delay(1000);
   }
  }

  Serial.println("Good to go!");
}

void loop() {
  time = millis();

  if(CAN_MSGAVAIL == CAN0.checkReceive() && line < 10000) {         // Check to see
whether data is read
   CAN0.readMsgBufID(&ID, &len, buf);    // Read data

//Add this line back in if you want to filter traffic    if(ID == 1201) { //39
    line = line + 1;

    Serial.print(ID,HEX); // Output HEX Header
    Serial.print("\t");

    for(int i = 0; i<len; i++) { // Output 8 Bytes of data in Dec
     Serial.print(buf[i]);
     Serial.print("\t");
    }

    Serial.print(time); // Timestamp
    Serial.print("\t");
    Serial.println(line); // Line Number
//   }
  }
  delay(10);
}
```

Appendix G

- **CAN Read Demo for the Spark Fun CAN Bus Shield.**

Written by Stephen McCoy.

Original tutorial available here: http://www.instructables.com/id/CAN-Bus-Sniffing-and-Broadcasting-with-Arduino

Distributed as-is; no warranty is given.

```
***************************************************************************/

#include <Canbus.h>

#include <defaults.h>

#include <global.h>

#include <mcp2515.h>

#include <mcp2515_defs.h>


//********************************Setup Loop*********************************//


void setup() {

  Serial.begin(9600); // For debug use

  Serial.println("CAN Read - Testing receival of CAN Bus message");

  delay(1000);
```

```
 if(Canbus.init(CANSPEED_500))  //Initialise MCP2515 CAN controller at the specified speed
   Serial.println("CAN Init ok");
 else
   Serial.println("Can't init CAN");

 delay(1000);
}

//*********************************Main Loop*********************************//

void loop(){

 tCAN message;
if (mcp2515_check_message())
         {
   if (mcp2515_get_message(&message))
         {
      //if(message.id == 0x620 and message.data[2] == 0xFF)  //uncomment when you want to
filter
         //{

           Serial.print("ID: ");
           Serial.print(message.id,HEX);
```

```
    Serial.print(", ");
    Serial.print("Data: ");
    Serial.print(message.header.length,DEC);
    for(int i=0;i<message.header.length;i++)
    {
      Serial.print(message.data[i],HEX);
      Serial.print(" ");
    }
    Serial.println("");
  //}
 }}

}
```

Appendix H

- **GPS code longitude and latitude (Arduino C language)**

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 9, TXPin = 3;
static const uint32_t GPSBaud = 9600;
// The TinyGPS++ object
TinyGPSPlus gps;
// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
void setup(){
Serial.begin(9600);
ss.begin(GPSBaud);
}
void loop(){
// This sketch displays information every time a new sentence is correctly encoded.
while (ss.available() > 0){
gps.encode(ss.read());
if (gps.location.isUpdated()){
Serial.print("Latitude= ");
Serial.print(gps.location.lat(), 6);
Serial.print(" Longitude= ");
Serial.println(gps.location.lng(), 6);
 }
 }
}
```

Appendix I

- **Potentiometer multiturn code (Arduino C language)**

```
float floatMap(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin A0:
  int analogValue = analogRead(A0);
Serial.print(analogValue );
Serial.print("\n" );
  // Rescale to potentiometer's voltage (from 0V to 5V):
  float steering_angle = floatMap(analogValue, 0, 1023, -780, 780);//mapping max steering
and min steering
Serial.print(steering_angle );
Serial.print("\n" );
 // delay(1000);
}
```