



**POLITECNICO  
DI TORINO**

**DEPARTMENT OF MECHANICAL AND  
AEROSPACE ENGINEERING**

**(DIMEAS)**

A thesis submitted to university in partial fulfillment of Master of Science in Automotive engineering

# EXPERIMENT SETUP DESIGN FOR NON-DIMENSIONAL VEHICLE DYNAMICS STUDIES

Candidate

Tholgappian Murugesan

Supervisors

Prof. Nicola Amati

Prof. Hormoz Marzbani

Turin, Italy

October 2019

## ABSTRACT

The aim of this work is to design and develop a complete setup for experimenting non-dimensional vehicle dynamics and validations over various control strategies and maneuvers for road cars. The target setup consists of an RC (Radio-Control) vehicle equipped with sensors capable of measuring real-time vehicle responses such as accelerations and yaw rate, and also applying autonomous steer-by-wire commands to the vehicle. Hardware network is built upon Arduino platform including an IMU sensor, Arduino microcontroller, and Wi-Fi modules. The key idea to design the experiment setup is derived from the dynamic similarity relationships between a real system (the prototype) and a scaled version (the model) using the science of dimensional analysis. The setup connects the dynamic responses of scaled vehicles to a 1:1 vehicle with a mathematical theorem called Buckingham Pi-theorem. This document contains the motivations, explanations, significance, procedures of the works. The step by step development of each part of the setup is clearly explained including formulation of Pi parameters, Arduino program codes, Steering control by wire, PID controller implementation, response logging, response post processing, setup installation in the vehicle and validation of the setup etc., Various constraints which were faced during the development process are documented under each topic.

## DECLARATION

I, Tholgappian Murugesan (S239209), do hereby declare that the contents of this thesis being submitted as a partial fulfillment of the Master of Science in Automotive Engineering, at Politecnico di Torino is original work of mine carried at Mechanical / Automotive department of Royal Melbourne University of Technology, Australia with the help and support of my supervisors. I confirm that the I have not used other

-----  
Tholgappian Murugesan  
(Candidate)

Certified that the above statement made by the student is true to the best of my knowledge

-----  
Prof. Nicola Amati  
(Tutor / Supervisor)

Place: Torino Italy

Date:

## **ACKNOWLEDGEMENT**

I would like to express my deep gratitude to Professor Nicola Amati, Professor Hormoz Marzbani, my research supervisors for giving me this amazing opportunity to work on this project, patient guidance and encouragement. I would particularly thank Mr. Sina Milani for his invaluable assistance, encouragement and useful critiques throughout the project.

I am particularly grateful to Politecnico di Torino for shaping me technically and personally during my course.

My special thanks to RMIT University staffs for supporting and providing laboratory equipment which really helped in taking the project faster and making my stay fruitful.

Finally, I wish to thank my mother for her constant support and my brothers Sarath, Gowrish and Anuj who made home, far from home for me.

# CONTENTS

CHAPTER 1 INTRODUCTION.....	9
1.1    MOTIVATION FOR THE RESEARCH.....	10
1.2    LITERATURE REVIEW.....	10
1.2.1 SCALED VEHICLES.....	10
1.2.2 VEHICLE PLANAR DYNAMICS THEORY .....	12
1.2.3 BICYCLE MODEL .....	13
CHAPTER 2 R/C CAR.....	14
2.1 TEST TYPE.....	14
2.2 TYPE OF VEHICLE .....	14
2.2.1 TRAXXASS X 01 SPECIFICATIONS.....	18
2.2.2 FEATURES.....	18
2.2.3 REASONS TO CHOOSE TRAXXAS X 01 .....	18
2.3 TEST STRATEGY .....	19
2.4 SUSPENSION .....	19
2.5. STEERING, TRACTION & RECIEVER .....	22
CHAPTER 3 DEVELOPMENT OF DATA ACQUISITION SYSTEM (DAS).....	24
3.1 SENSORS .....	25
3.1.1 MPU-6050.....	25
3.1.2. I2C.....	26
3.1.3 SERIAL UART PORTS.....	26
3.1.4 SPI COMMUNICATION INTERFACE .....	27
3.1.5 I2C INTERFACE .....	27
3.2 ARDUINO .....	28
3.2.1 ARDUINO DUE.....	28
3.3 WI-FI SHIELD .....	30
3.3.1 Wi-Fi.....	30
3.3.2 NODE MCU 1.0.....	30
3.3.3 TCP – TRANSFER CONTROL PROTOCOL.....	31
3.4 PC USER INTERFACE.....	31
3.4.1 SCRIPT COMMUNICATOR .....	31

3.4.2 MATLAB .....	32
CHAPTER 4 PROGRAMMING THE SETUP .....	33
4.1.1 ARDUINO IDE .....	33
4.2 ACQUIRING DATA FROM SENSOR .....	34
4.3 DATA TRANSMISSION .....	37
4.3.1 SETTING UP Wi-Fi: TRANSMITTER .....	39
4.3.2 SETTING UP Wi-Fi: RECEIVER .....	40
4.4 STEERING MOTOR CONTROL .....	41
CHAPTER 5 SETUP VALIDATION .....	43
5.1 YAW MOMENT OF INERTIA ESTIMATION .....	44
5.2 CORNERING STIFFNESS ESTIMATION .....	47
5.3 VALIDATING WITH BICYCLE MODEL .....	52
5.3.1 CURVATURE GAIN .....	52
5.3.2 UNDERSTEERING COEFFICIENT .....	52
5.4.3 OTHER RESPONSES- STEADY STATE .....	52
5.4.4 OTHER RESPONSES .....	55
CHAPTER 6 PI ANALYSIS .....	57
CHAPTER 7 EXAMPLES .....	60
7.1.1 PID CONTROL .....	60
7.1.2 PID TUNING CIR/CUIT .....	63
CHAPTER 8 SUPPORT STRUCTURE .....	66
CHAPTER 9 CONCLUSION .....	68
REFERENCES .....	71

FIGURE 1 BICYCLE MODEL – CORNERING (JAZAR, 2017) .....	12
FIGURE 2 BICYCLE MODEL (JAZAR, 2017) .....	13
FIGURE 3 TRAXXAS X01 FRONT VIEW.....	15
FIGURE 4 TRAXXAS X01 SIDE VIEW .....	15
FIGURE 5 TRAXXAS X01 TOP VIEW .....	16
FIGURE 6 TRAXXAS X01 BODY (TRAXXAS, 2019) .....	16
FIGURE 7 TRAXXAS X01 DISMANTLED .....	17
FIGURE 8 TRAXXAS X01 TIRES.....	17
FIGURE 9 SUSPENSIONS FRONT AND REAR (TRAXXAS, 2019).....	19
FIGURE 10 SUSPENSION RODS CATIA MODEL AND DRAWING .....	20
FIGURE 11 SUSPENSIONS ASSEMBLED .....	20
FIGURE 12 SUSPENSION RODS ASSEMBLED .....	21
FIGURE 13 STEERING SERVO MOTOR(TRAXXAS, 2019) .....	22
FIGURE 14 STEERING MECHANISM - BELL CRANK.....	22
FIGURE 15 COMPLETE MODEL WIRING DIAGRAM (TRAXXAS, 2019) .....	23
FIGURE 16 MPU 5060 .....	26
FIGURE 17 I2C BUS TOPOLOGY.....	26
FIGURE 18 UART BUS TOPOLOGY .....	27
FIGURE 19 SPI BUS TOPOLOGY .....	27
FIGURE 20 ARDUINO DUE(ARDUINO, 2019).....	29
FIGURE 21 ARDUINO DUE PIN OUT (ARDUINO, 2019).....	29
FIGURE 22 ESP 12 PIN OUT (MCU, 2019).....	30
FIGURE 23 DATA LOGGING CONSOLE .....	31
FIGURE 24 ARDUINO IDE (ARDUINO, 2019).....	34
FIGURE 25 MPU 6050, ARDUINO DUE WIRING DIAGRAM .....	35
FIGURE 26 ARDUINO DUE, ESP 12, MPU 6050 WIRING DIAGRAM.....	38
FIGURE 27 STEERING SERVO, ARDUINO DUE WIRING DIAGRAM .....	41
FIGURE 28 MASS MOMENT OF INERTIA ABOUT Z AXIS – BIFILAR PENDULUM TECHNIQUE .....	44
FIGURE 29 BIFILAR PENDULUM ASSEMBLED .....	45
FIGURE 30 LONGITUDINAL ACCELERATION VS TIME .....	48
FIGURE 31 YAW RATE VS TIME.....	49
FIGURE 32 LATERAL ACCELERATION VS TIME .....	49
FIGURE 33 YAW RATE RESPONSE VS LONGITUDINAL VELOCITY .....	53
FIGURE 34 LATERAL VELOCITY RESPONSE VS LONGITUDINAL VELOCITY .....	53
FIGURE 35 VEHICLE SLIP VS LONGITUDINAL VELOCITY.....	54
FIGURE 36 RADIUS OF CURVATURE VS LONGITUDINAL VELOCITY.....	54
FIGURE 37 SIMULINK MODEL .....	55
FIGURE 39 YAW RATE RESPONSE VS TIME .....	56
FIGURE 38 LATERAL VELOCITY RESPONSE VS TIME.....	56
FIGURE 40 ARDUINO DUE, STEERING SERVO , POTENTIOMETERS WIRING DIAGRAM - PID TUNING .....	65
FIGURE 41 SUPPORT STRUCTURE CAD MODEL.....	66
FIGURE 42 TECHNICAL 2D DIAGRAM - SUPPORT STRUCTURE (ALL DIMENSIONS IN MM).....	67
FIGURE 43 COMPLETE SETUP WITH ARDUINO DUE, WI-FI MODULES, SENSOR , POWER SUPPLY, STEERING SERVO WITH SUPPORT STRUCTURE ASSEMBLED ON THE R/C CAR.....	67



# CHAPTER 1 INTRODUCTION

Vehicle dynamics study to be carried out in a real-time full-sized car requires considerable resources and extreme safety precautions. The aim of this work is to create an experiment setup which can retrieve and scale the vehicle dynamics data from a scaled car (R/C car) to a full-sized car. This work facilitates in matching the planar dynamic performance of two parametrically equivalent vehicles having varied scales. The limitations and boundaries in using scaled vehicles, an overview of model parameters and concept of reviewing dynamic similitude in domain of interest are documented in this proposal. In recent years, the idea of scaled simulation and experimentation has been widely spread. These types of studies are extensively used in aerospace industries due to the inability to do high frequency testing indoor. The experimental setup at the end of the project is capable in accelerating vehicle control validation processes with data logging, strategy and maneuver application in real time. Dynamic similitude research helps to study about the systems and their behaviors that are difficult to study in their original size and normal operating environment. Normally, such research uses scaled models that are dynamically similar to a system which are much larger than the model. Using similitude theory, the dynamics of a system can be studied in terms of dimensionless parameters. An important contribution in this development belongs to Buckingham, who invented a theorem, called PI- theorem, that can be used to study the properties of scaled systems and matching with full-sized systems.

## 1.1 MOTIVATION FOR THE RESEARCH

There is a strong motivation to study the vehicle dynamics and its control. For automotive industries it is difficult to justify the euros and time spent on vehicle on road testing and validation. A question central to vehicle on-road testing is that is it possible to do the testing with low cost and time and even indoors. Because, each vehicle validation requires thousands of dollars, hours and man power. The question is valid and it depends on what technologies and theories we have, to perform those.

## 1.2 LITERATURE REVIEW

### 1.2.1 SCALED VEHICLES

From a research point, the first advantage of using scale vehicles over full-sized vehicles is cost. Using a full-sized vehicle testing is considerably expensive to most institutions and organizations. Few research companies conducting vehicle testing has very smaller grants to conduct their testing tasks, and most of the capital goes simply to development of infrastructures. The cost of the entire test setup is around \$3000, which includes the cost of an R/C car, the transmitter system, and sensors including spare parts. For comparison, it is estimated that the cost of using a full-sized vehicle for testing is from \$50,000 and more. Normal research institutions cannot withstand this price tag. In comparison to scaled testing, full-sized vehicles testing spends majority of the money on equipment, road usage and taxes. But this realistic effect has high cost intensity. Full-sized autonomous vehicles at research universities are rarely used in real testing which pushes vehicles to its performance limits. An additional advantage of time using a scale vehicle system, the time required to alter a scaled vehicle is minimal compared to full-size vehicle. In terms of commercial vehicles, a primary cost falls on the start-up & turn-around time needed for idea. While an entire scaled vehicle can be built "from scratch" in a short time provided all the parts are available with small adjustments. The durability of the R/C cars and the downtime due to vehicle breakage is small. Treadmill system avoids scheduling of roadway usage with a constant road condition, no traffics, various road surfaces can be simulated, and real-time evaluation of vehicle performance can be conducted at any time leaving testing safer, consistent and repeatable which in turn helps to do aggressive testing. Frequency responses requiring a full day of testing can be conducted continuously on the vehicle using swept-sine techniques eliminating expensive GPS systems as in full size vehicles. Finally, scaled vehicles are simpler and safer than full-sized vehicles to test. No drivers or no pedestrians are at risk and there is no worry about traffic as the computer drives the car.

There are two distinct ways to study vehicle dynamics. The first is to establish the equations and conduct a simulation-based study, and assume that the vehicle follows the theoretical differential equations and the parameters used. The resulting controllers are in general solved in terms of these parameters. Fundamental to the controller is the assumption that the unmodeled dynamics are insignificant enough to be compensated by feedback. The second methodology is to test a full-sized vehicle and conduct semi-empirical analysis (black-box approach) of the full-sized vehicles using input/output modeling. The use of scaled vehicles

generally allows both approaches. An advantage of scaled vehicles is that they encourage both methodologies: the theoretical analysis is used to validate scale dynamic similitude, but the use of a physical model ensures that critical dynamics will not likely be neglected. Before simulation-based modeling, vehicle testing was done either on scaled or full-sized vehicles.

The use of scale vehicles was preferred because of cost reasons, and consequently there was extensive use of scaled vehicles through the mid-1960's due to cold-war military research and space-race sponsored studies. The use of scale vehicles has extended to automobiles in the areas of crash reconstruction, vehicle-soil interaction (tire forces), suspension analysis and dynamics, and roll dynamics. Scale testing of amphibious vehicles was conducted extensively at the Army's Land Locomotion Laboratory in Detroit (Bekker). Extensive scale model studies, conducted in 1960's that specifically examined tire forces of scaled models and their scaling, and showed that Scaled testing is one of the best ways to determine the vehicle turning radius(Bekker)

In the field of automotive accident reconstruction, detailed analysis was conducted examining vehicle crashes of scaled ones before computer simulation-based studies. Dynamic similitude between scale and full-sized vehicles having same forces in crash was derived and examined. Experiments demonstrated agreement between scale and full-sized automobiles in non-crash dynamics of automobile accidents (Emori, 1969)

Scaled vehicles have been used to study dynamic behaviors of complex multi-body systems. Back in 1930's, Huber and Dietz (1937) used a treadmill to conduct experimental work with small-scale models of tractor-trailer combinations (Bekker). This method of using a treadmill to study vehicle dynamics was also used by to study trailer stability (Bekker). A study during the 1960's by the U.S. Army Land Locomotion Laboratory was conducted to examine the turning behavior of two-unit, tracked, articulated transportation using scale vehicles(Bekker). This study includes both nonlinear and steady-state turning for different vehicle configurations. A validation of the turning radius was conducted for tracked vehicles using scale vehicles in 1950's by (Nuttal, 1951). For very complex multi-mode vehicle behavior like to tune the suspension systems of vehicles over very rough terrain would limit the use of scale vehicles since the obstacles are larger than the vehicle itself.

One of the most significant uses of scaled vehicles for vehicle dynamics studies has been introduced by Brennan and Alleyne (2001a) and Lapapong, Gupta, Callejas, and Brennan (2009) where the use of non-dimensional parameters based on Buckingham (1914) drastically reduced the parametric uncertainties. The parameters of interest may be large in number but number of sensors which can be equipped in a scaled car is limited since weight, volume of the sensors, measurement accuracy, processing power and time has larger significance. In these cases, the non-dimensional parameters trim the number of variables required. The physical parameters essential to study vehicle planar dynamics are mass, wheel base, vehicle yaw moment of inertia, wheel track, and cornering stiffness, while the variables to be measured/estimated are yaw rate, longitudinal and lateral accelerations, vehicle side-slip angle, wheel and body velocities. Due to non-linearity of tires, their cornering stiffness has significant effects on vehicle modelling as well as vehicle dynamics as it changes with variations in roads and vertical load over it. Sierra, Tseng, Jain, and Peng (2006) work gives various methods in estimating cornering stiffness including a method without side slip angles and has explained the limitations and involved parameters. Lian, Zhao, Hu, and Tian (2015) explains

the estimation without side slip angle but with a nonlinear observer. Fujimoto and Takahashi (2006) provide three different ways to estimate by using state observer accompanied with yaw rate controller.

As another significant example of non-dimensional analysis, Polley and Alleyne (2004) used a scaled tire and measured the stiffness using a test bed and extrapolated it to a full-sized wheel using non-dimensional parameters. Brennan and Alleyne (2001b) effort to build Illinois Roadway Simulator demonstrated the use of control to change vehicle dynamics rom driver point of view and helped understanding the dynamic similitude between scaled and full-sized vehicles thus reducing the controller designs efforts. O'Brien, Piepmeier, Hoblet, Burns, and George (2004) took straight forward modifications on the parameters in studying the dynamics similitude. Kalinowski (2013) in his thesis gave a brief insight on how Arduino boards can be implemented in low power, low level vehicle controls.

### 1.2.2 VEHICLE PLANAR DYNAMICS THEORY

This section describes the vehicle planar dynamics from a theoretical approach. Firstly, the history of the modeling of vehicle dynamics is discussed. Rigid vehicle is assumed to act similar to a flat box moving on a horizontal surface. A rigid vehicle has a planar motion with three degrees of freedom that are: translation in the x and y directions, and a rotation about the z axis. The vehicle notations which were used throughout this document is then presented. The equations of motion are obtained by resolving the acceleration components for the vehicle in terms of a coordinate system centered on the moving vehicle. Forces on the tires are then discussed. Tire and body dynamics is then linearized to obtain the model called bicycle model. The steady-state solution, is then discussed. Transfer functions which relates vehicle input to output are then given, and general vehicle dynamic trends are then deduced from the functions.

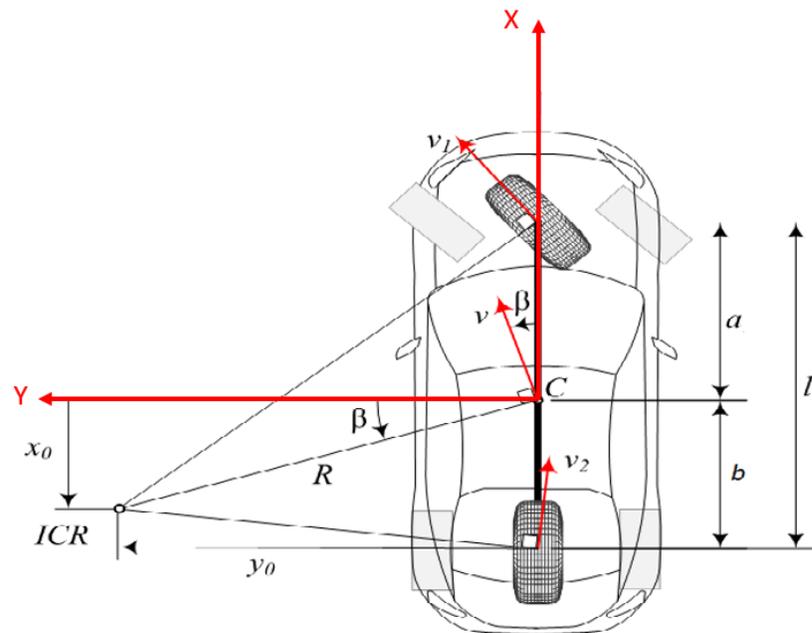


Figure 1 Bicycle model – Cornering (Jazar, 2017)

### 1.2.3 BICYCLE MODEL

Vehicle lateral dynamics has been studied since 1950. In order to describe roll, yaw, and lateral motions at a constant speed, a 3 DOF vehicle model developed by Segel (Segel, 1956). If the roll motions are ignored, a simpler model is obtained called bicycle model. Newton's equations of motions can also be applied to the bicycle model. The bicycle model has been widely used for basic level control due to its simplicity. For small angles and accelerations, the approximated Newtonian and full Newtonian methods give identical transfer functions.

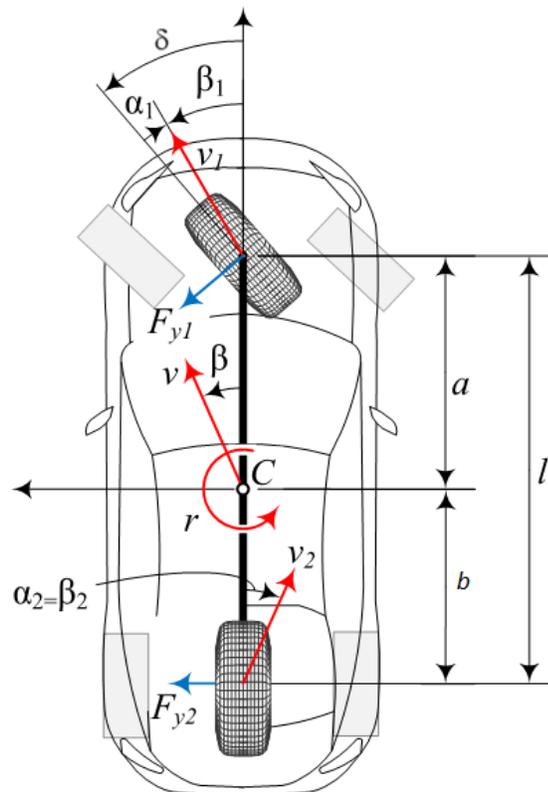


Figure 2 Bicycle model (Jazar, 2017)

## CHAPTER 2 R/C CAR

R/C car (Radio -Controlled) is a typical scaled car with electric drive controlled by a handheld radio controller at the user. The motto of this project is to apply non-dimensional analysis strategy which can be used for vehicle testing. So, the R/C car's features and dimensional ratios must resemble more of a real prototype car is preferable. The primary questions to answer are

1. What type of test
2. What type of vehicle
3. Where and how

### 2.1 TEST TYPE

As previously informed, the strategy so far is applicable for planar vehicle dynamics only. Moving complex towards the dynamics increases the complexity in using the analysis strategy. So, the test is Vehicle planar dynamics test in indoors. Concentrating on planar dynamics, the major affecting parameters are center of gravity location, mass, type of tires and type of traction.

### 2.2 TYPE OF VEHICLE

Caution is carried while choosing the scaled car for testing. The R/C car which more closely resemble full sized car in all aspects is desirable. The aspects include type of traction, tires, weight

distribution, location of COG, and overall dimensions of the car. The importance in stressing the points is to make sure that the scaled car gives close results to full sized car. Whereas the above aspects are closely related to vehicle planar dynamics. Upon close search on the market , the car with most preferable features was found. The car chosen was Traxxas X01.



*Figure 3 Traxxas X01 Front View*



*Figure 4 Traxxas X01 Side view*



Figure 5 Traxxas X01 Top View

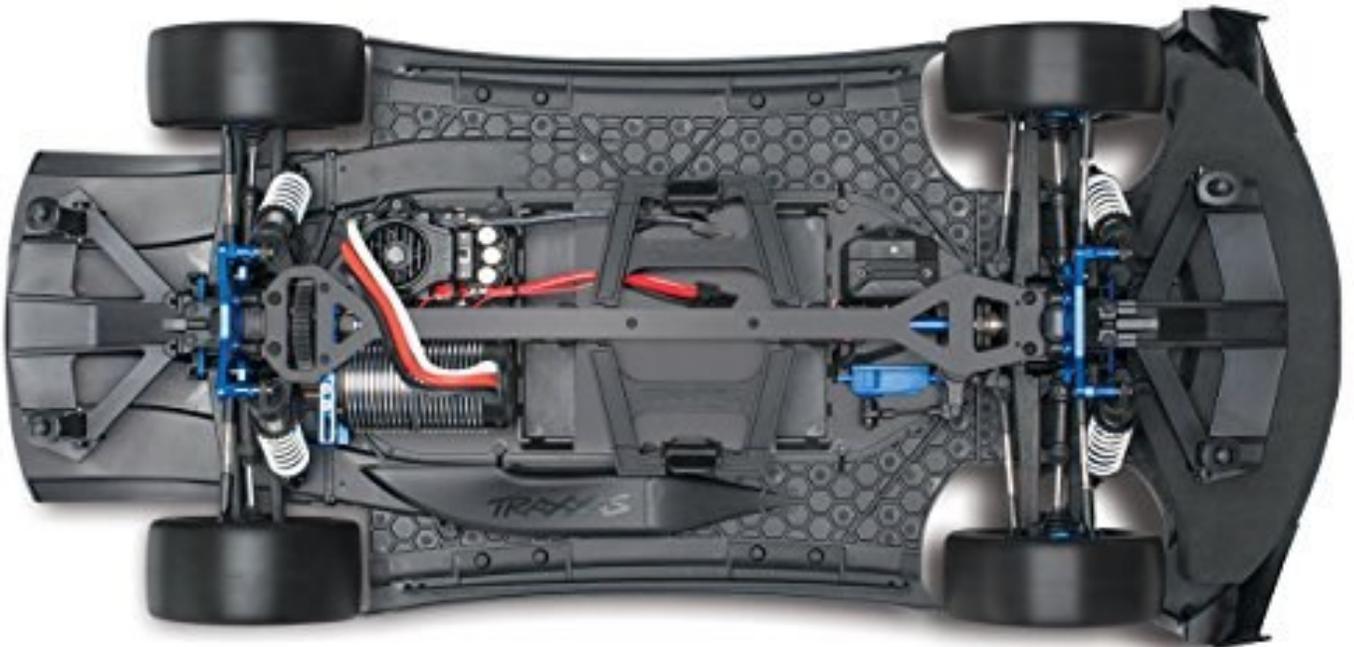


Figure 6 Traxxas X01 Body (Traxxas, 2019)



*Figure 7 Traxxas X01 Dismantled*



*Figure 8 Traxxas X01 Tires*

## 2.2.1 TRAXXASS X 01 SPECIFICATIONS

SPECIFICATIONS (Traxxas, 2019)	
Length:	686 mm
Front Track:	295 mm
Rear Track:	300 mm
Center Ground Clearance:	15 mm
Weight:	3.9 kg (w/o batteries)
Height (overall):	127.5 mm
Wheelbase:	404 mm
Front Shock Length:	83 mm
Rear Shock Length:	86 mm
Front Tires (pre-glued):	Belted slick (4.29" x 1.7")
Rear Tires (pre-glued):	Belted slick (4.29" x 2")
Front Wheels:	3.3" Split-Spoke™ (black-chrome)
Rear Wheels:	3.3" Split-Spoke™ (black-chrome)
Motor (electric):	Traxxas Big Block Brushless (1650 Kv)
Transmission:	Single-speed
Overall Drive Ratio:	9.36
Differential Type:	Sealed hardened steel bevel
Chassis Structure/Material:	3mm plate, dual plane, 6061-T6 aluminum
Brake Type:	Electronic
Drive System:	Shaft-driven 4WD
Transmitter:	TQi™ 2.4GHz Radio System with Traxxas Link Wireless Module
Battery Tray Dimensions:	155mm x 50mm x 29mm
Required Batteries & Charger:	Two LiPo batteries (LiPo, 3-Cell 5,000 mAh), 4 "AA" (transmitter), LiPo battery charger

## 2.2.2 FEATURES

- Top speed up to 100mph with Brushless motors
- Flat Underbody with molded tire rims, inserts resembling real race tires
- Inbuilt stability management system
- Variable sensitivity and trims of throttle and steering
- Integrated telemetry sensors for wheel RPM, Car battery voltage and temperature
- 2.4GHz Radio with Wireless Module
- Model memory and drive effects are accessible via the mobile App
- High torque servo for steering
- Almost zero latency in control response

## 2.2.3 REASONS TO CHOOSE TRAXXAS X 01

1. It is big, unlike other R/C cars, X01 is of 1:7 scale model
2. Tires are more like real race car tires
3. Overall dimensions can be scaled close to full sized car which helps in applying PI theorem
4. COG location is same like full sized car
5. Weight distribution is same like full sized car
6. Four-wheel drive can be changed to front/rear wheel drive easily

7. Steering and throttle is PWM controlled
8. Steering and throttle trims can be controlled via mobile app
9. Stability management can be switched off
10. Long lasting batteries favors long duration testing
11. Various test profiles can be fed easily through mobile app

## 2.3 TEST STRATEGY

The R/C car can be tested indoors as well as outdoors. While in testing outdoors the primary concern is amplitude of vibrations from the road which is too high for the small car. Proper filtering techniques should be applied before processing the data. Alternatively, the car can be tested indoors at low speed on a treadmill like many other researches.

Changes made on the car

1. Replaced suspension springs with solid rods
2. Removal of aerodynamics features like splitters etc.
3. Electronic Stability management is switched off
4. Steering unlinked from TQi controller and controlled by micro controller

## 2.4 SUSPENSION

The suspension came with built in R/C car (double-wishbone at front and rear) is same like full size car whose suspension parameter like end stop, spring stiffness and camber can be altered by user. But since we are interested in just the planar dynamics the effects of the suspensions should be removed or not considered. Hence the suspensions are removed and plastic rod is replaced. The plastic rod has the same dimensions of the suspensions with no spring and damping. The model is designed using CATIA by following the dimensions given in the user manual of the R/C car. Later the model is 3D printed in Zotrax M200 3D printer available in RMIT Bundoora facility.



Figure 9 Suspensions Front and Rear (Traxxas, 2019)

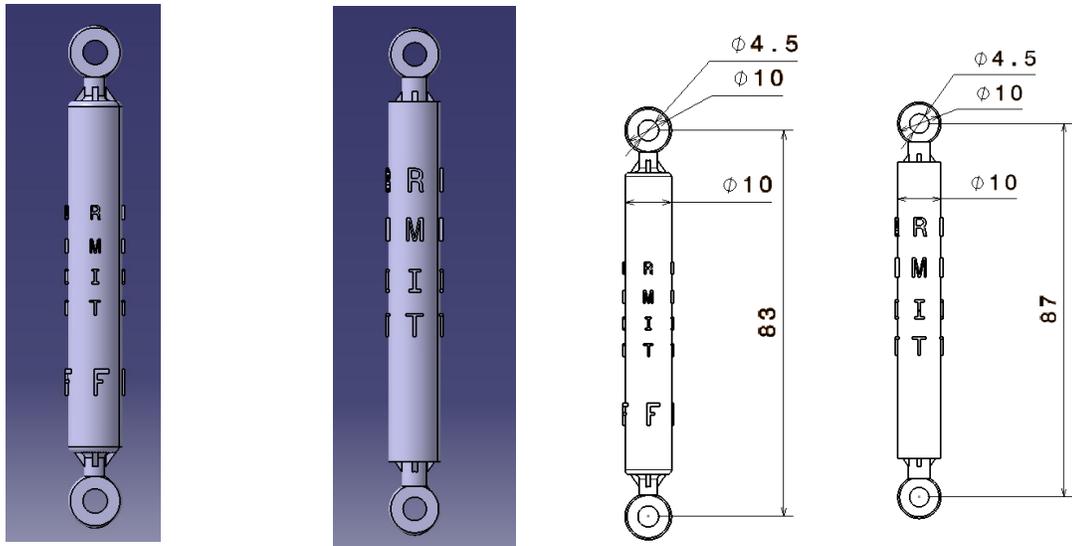
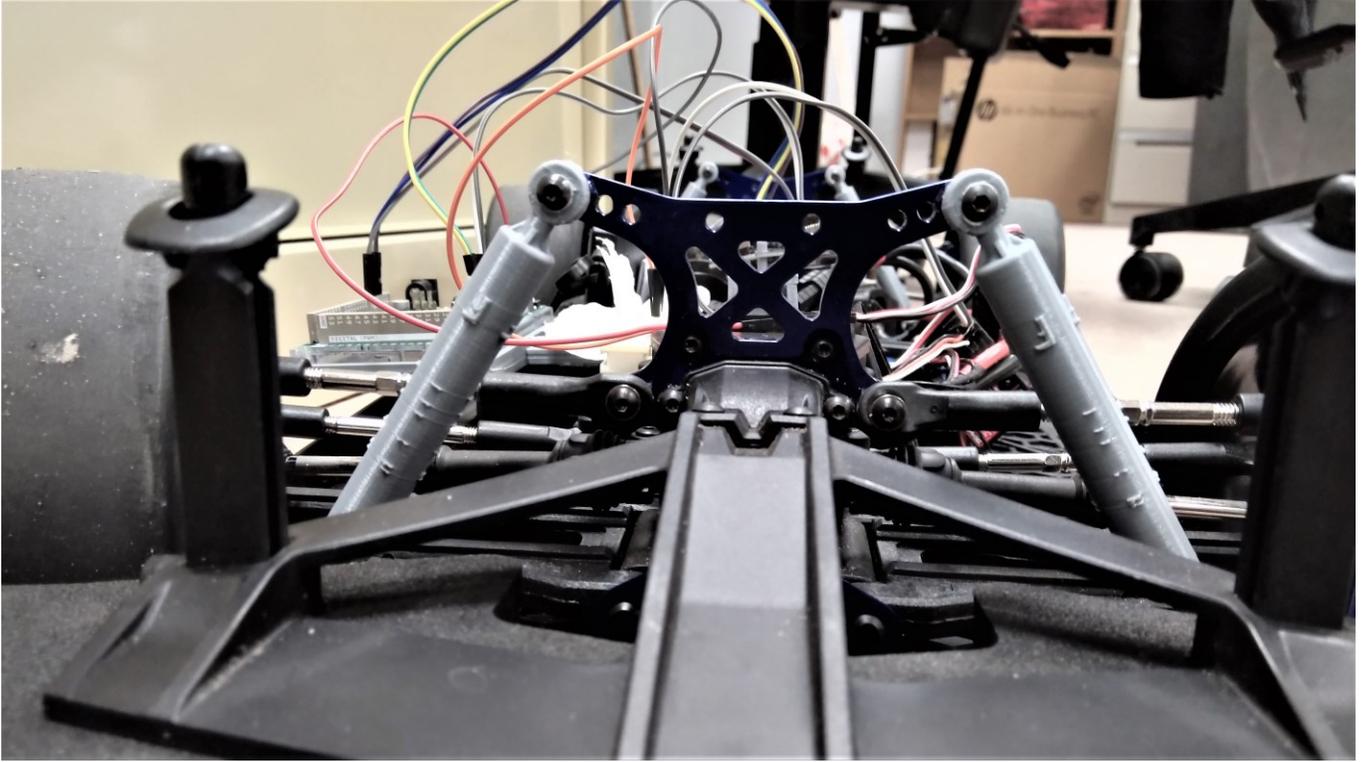


Figure 10 Suspension rods CATIA model and Drawing



Figure 11 Suspensions Assembled



*Figure 12 Suspension rods Assembled*

## 2.5. STEERING, TRACTION & RECIEVER

The steering mechanism implemented by the manufacturer in the R/C car is Bell crank Mechanism. It obeys Ackermann kinematic steering principle. The controlling the steering may be useful in performing steady steer tests. The steering is electrically powered by the batteries in the R/C car and the crank motion is given by a servo motor. The motor is capable of providing 9Kg/cm of torque. As this is a digital servo motor, it can be controlled by PWM. The voltage required to drive the servo motor is 14V, but the control pulse can be fed from a 5V source also. The servo motor has three connections.

Red – V+ : Black – GND : White – Pulse/ PWM



Figure 13 Steering servo motor(Traxxas, 2019)

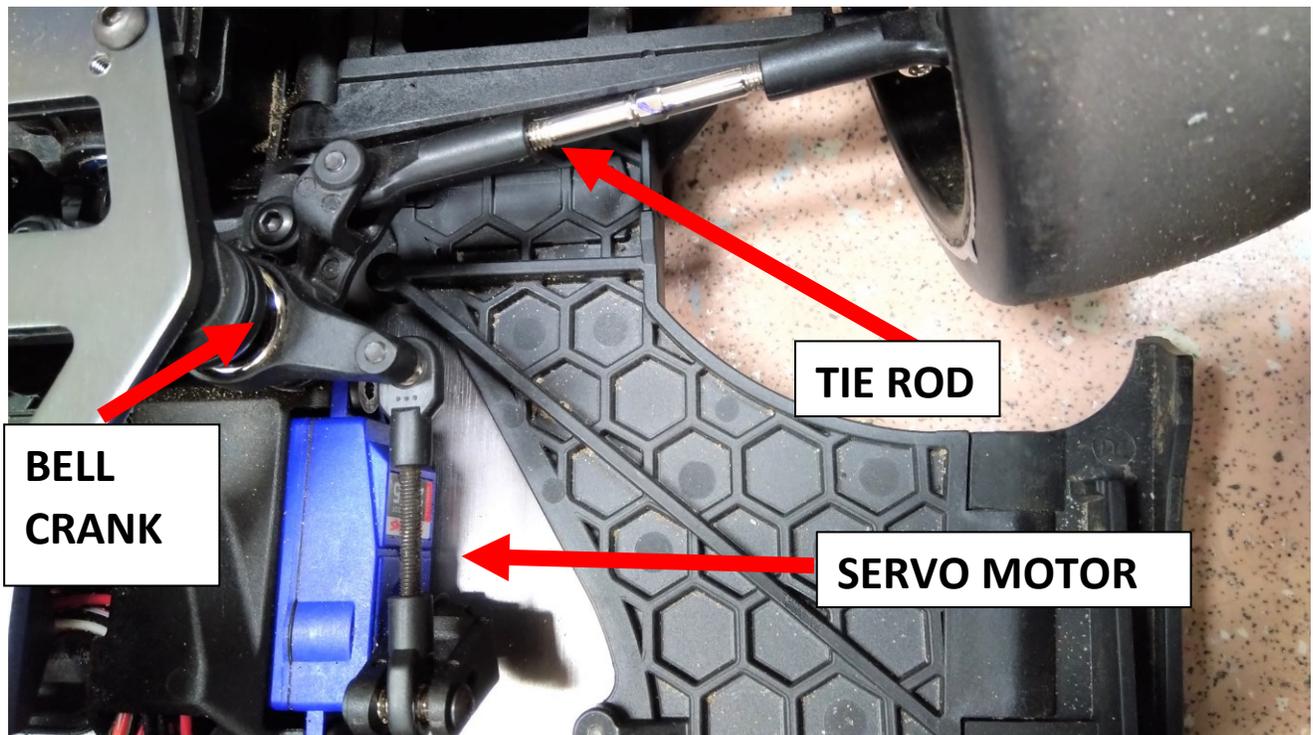


Figure 14 Steering Mechanism - Bell crank

The traction is provided by 1650 Kv brushless motors and operates at 14V. The control input from the transmitter is received through the antenna available at the car. The RPM feedback is given to Electronic Speed Control to achieve desired RPM. The complete model wiring diagram with pin out of the receiver is shown in the below diagram.

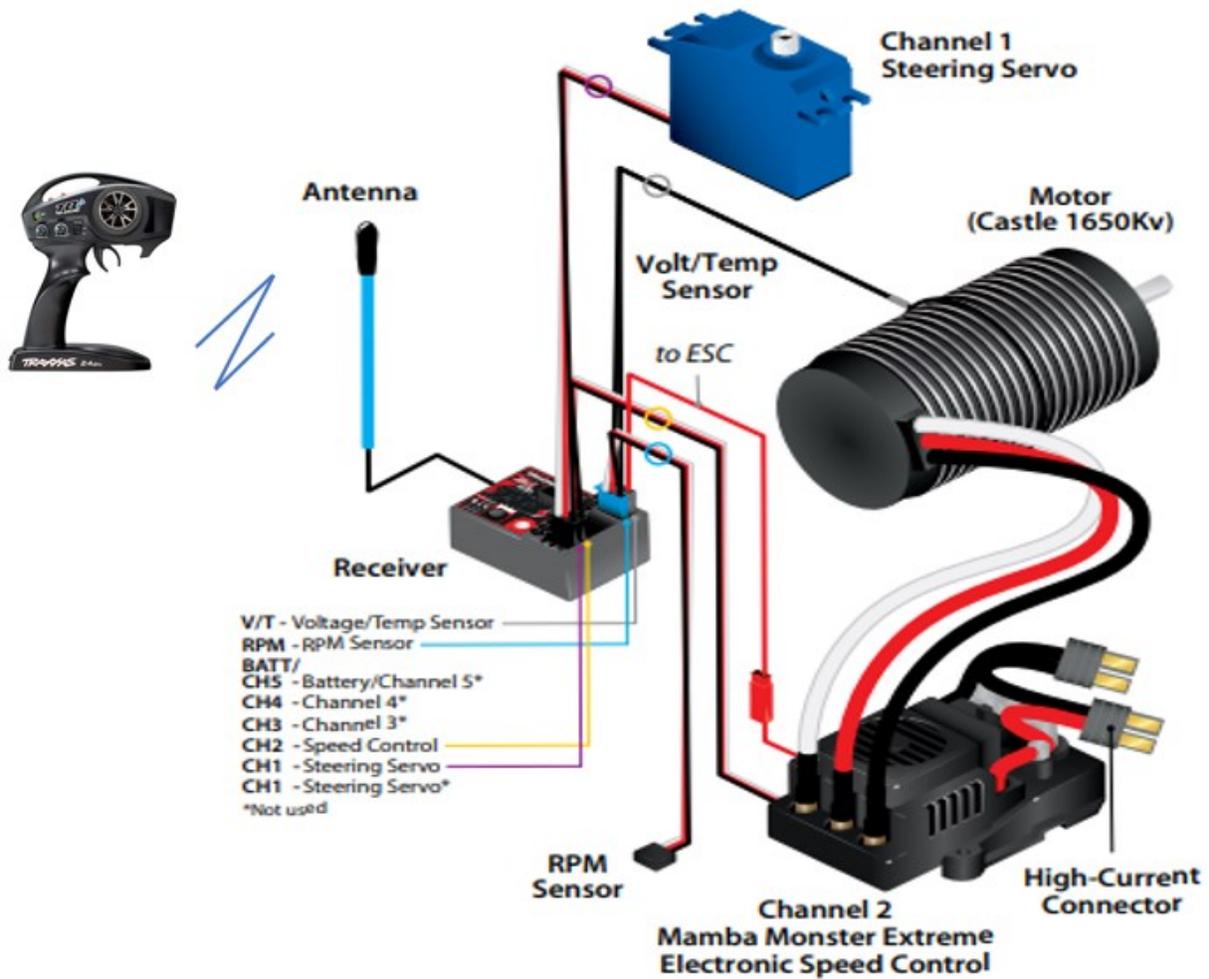


Figure 15 Complete model wiring diagram (Traxxas, 2019)

## CHAPTER 3 DEVELOPMENT OF DATA ACQUISITION SYSTEM (DAS)

The connection architecture suitable for this setup is Local Interconnect Network [LIN].

The Local Interconnect Network (LIN) is a standard low-cost bus, low-end multiplex communication commonly seen in modern automotive networks. LIN is relatively inexpensive communication setup made using the standard serial universal asynchronous receiver/transmitter (UART) embedded into most modern low-cost 8-bit microcontrollers such as Arduino. This architecture provides cost-efficient communication where complex functions and versatility is not required.

Modern automotive networks use a combination of LIN for low-cost applications primarily in body electronics mainly while communicating with sensors. The LIN bus uses a master/slave approach that comprises a LIN master and one or more LIN slaves exactly what is need in this project to read data from sensor.

Things needed

1. Sensors
2. Arduino
3. WI-FI Shield
4. PC User Interface

### 3.1 SENSORS

The objective of the project is constrained to planar dynamics where the number of parameters involved is significantly less. The data required from the car, should be read by the sensor are Acceleration about three axes and angular velocity about the same three axes. There are low cost micro electromechanical sensors available on the market which can give accuracy about  $\pm 16g$  and  $\pm 2000dps$  which satisfies the requirement for the project. Available MEMS in the market is given below:

*Table 1 Sensors comparison*

	ACC	GYRO	INTERFACE	FREQ	OUTPUT	ANALOG/ DIGITAL	RANGE	V	COST
<b>MPU 6050</b>	Yes	Yes	I2C	1KHz	16	Digital	Max $\pm 16g$ / 2000dps	3.3V	\$\$
<b>ADXL 345</b>	Yes	No	I2C / SPI	2MHz	16	Digital	Max $\pm 16g$	3.3V	\$\$\$
<b>LIS3DH</b>	Yes	No	I2C	5KHz	10	Digital /Analog	Max $\pm 16g$	3.3V	\$

Out of these, MPU 6050 has combined Accelerometer and gyro meter while the others not. Sampling frequency and cost also looks convincing.

#### 3.1.1 MPU-6050

The MPU-6050 is a 6-axis IMU with both 3-axis gyro and a 3-axis accelerometer on the same die which works at low power. The 6 DOF sensor has a low noise 3.3v regulator and pull-up resistors for the I2C bus. So, it's readily available to directly hook up the sensor with the microcontrollers. Arduino SAM I2C library and MPU 6050 library make it easy to drive the sensor and collect the acceleration and angular velocity data. The device has integrated Motion Fusion algorithm to allow access to the meters via I2C bus, allowing the devices to collect a full set of sensor data without intervening system processor.

#### SPECIFICATIONS

- Working voltage: 3-5v
- I2C Digital output
- 3-axis angular velocity with ranges of  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , and  $\pm 2000dps$
- 3-axis accelerometer with a range of  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$  and  $\pm 16g$
- Dimensions: 14 x 21mm



Figure 16 MPU 5060

### 3.1.2. I2C

The communication bus available for MPU 6050 is I2C bus. However, other communications types are available as well, there are specific reason which makes MPU 6050 shine above others.

The Inter-integrated Circuit (I<sup>2</sup>C) Protocol is a protocol designed to allow many slave devices to communicate with a single master device. It is for short distance communication and requires only two wires for data transfer reducing the complexity in wiring. To figure out why one might want to prefer I<sup>2</sup>C, initial step is to compare it to the other available options.

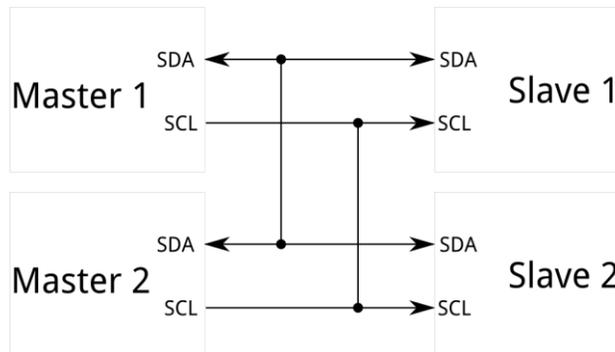


Figure 17 I2C Bus Topology

### 3.1.3 SERIAL UART PORTS

Serial ports are **asynchronous**, so devices should have nearly same data rate and same clocks. If those are not matching the receiver may receive garbage data. UART ports requires hardware overhead software on both sides for control which is relatively complex and difficult to implement.

UART can communicate between only two devices. Finally, data rate is an issue. This communication requires 10 bits of transmission time for each 8 bits of data sent, which eats into the data rate. Most UART devices only support fixed baud rates.

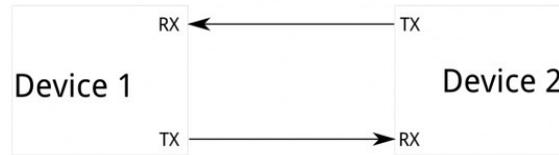


Figure 18 UART Bus Topology

### 3.1.4 SPI COMMUNICATION INTERFACE

The most obvious drawback of SPI is the number of wires required. Just to connect one master to one slave device with an SPI communication bus requires four wire lines which makes each extra slave requires one extra I/O pin on the master. SPI only allows one master on the bus, and arbitrary number of slaves (subject only to the drive capability of the devices connected to the bus and the number of chips select pins available).

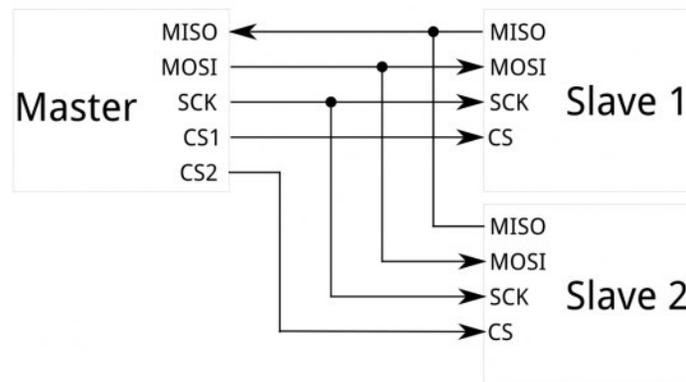


Figure 19 SPI Bus topology

### 3.1.5 I2C INTERFACE

Considering the major drawback of the other two communication types, such as sync problems, multiplexing many devices, complexity in wiring, data rate etc, I2C make a clear get through.

I<sup>2</sup>C requires only two wires which can support up to 1008 slave devices. Unlike in SPI, I<sup>2</sup>C can support a multi-master system, allowing more than one master to communicate with all slave devices which is desired here. Data rates of I<sup>2</sup>C devices is at 100kHz or 400kHz. There is some overhead with I<sup>2</sup>C; for every 8 bits of data to be sent, one extra bit of ACK/NACK must be transmitted ensuring the delivery of the data.

Each I<sup>2</sup>C bus consists of two signals: SCL and SDA. SCL is the clock signal, and SDA is the data signal. The clock signal is always generated by the master. There can be no bus contention, eliminating the potential for damage to the drivers or excessive power dissipation in the system.

## 3.2 ARDUINO

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits. The table 3 compares the list of Arduino microcontrollers available in the market.

Table 2 Arduino Comparison (Arduino, 2019)

Name	Processor	Operating/ Input	CPU Speed	Analog In/Out	Digital IO/PWM	Flash [kB]	UAR T
		Voltage					
<a href="#"><u>Mega 2560</u></a>	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	256	4
		5 V / 5-12 V	16 MHz				
<a href="#"><u>Uno</u></a>	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14	32	1
<a href="#"><u>Zero</u></a>	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6	14	256	2
<a href="#"><u>Due</u></a>	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12	54	512	4
<a href="#"><u>Mega ADK</u></a>	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	256	4

### 3.2.1 ARDUINO DUE

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. This Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 84 MHz clock, an USB OTG capable connection, 2 Digital to analog pins , 2 TWI, a power jack, 4 UART hardware serial ports, a reset button and an erase button which is a lot compared to other boards.

Arduino Due board runs at 3.3V while other boards uses 5V. The I/O pins can tolerate only 3.3V. Applying voltages higher than 3.3V to any I/O pin could damage the board which in turn helps to save power. Just by simply connecting it to a computer with a micro-USB cable is enough to get started. The board is compatible with all Arduino shields that work at 3.3V. The figures and pinout diagram of Arduino Due is shown below.



### 3.3 WI-FI SHIELD

#### 3.3.1 Wi-Fi

A wireless network uses radio waves and is a lot like two-way radio communication. A wireless adapter can translate data into a radio signal in terms of 0s and 1s and transmits it. The router receives the signal and converts it. The router transfers the data to the other network nodes using a physical, wired Ethernet connection. The process can also work in reverse, while router receiving information from the Internet, converting it and sending it to the computer's wireless adapter. Wi-Fi radios have a few notable differences from other radios like they can transmit at frequencies of 2.4 GHz or 5 GHz. This higher frequency means more signal toggles within a time which means more data.

Wi-Fi radios can transmit on three frequency bands. They can "frequency hop" rapidly between the different bands to ensure connectivity. Frequency hopping helps in reducing interference and lets multiple devices use the same wireless connection simultaneously.

#### 3.3.2 NODE MCU 1.0

NodeMCU is an open source IoT platform with ESP8266 Wi-Fi SoC firmware loaded.

NodeMCU 1.0 consists of an ESP-12E on a board which is a kind of microcontroller platform. So, this module can create a Wi-Fi network as well as can be used to control miniature thing on the go.

It also has inbuilt voltage regulator and a USB interface. The ESP8266 is a Wi-Fi chip with full TCP/IP stack and microcontroller at low cost.

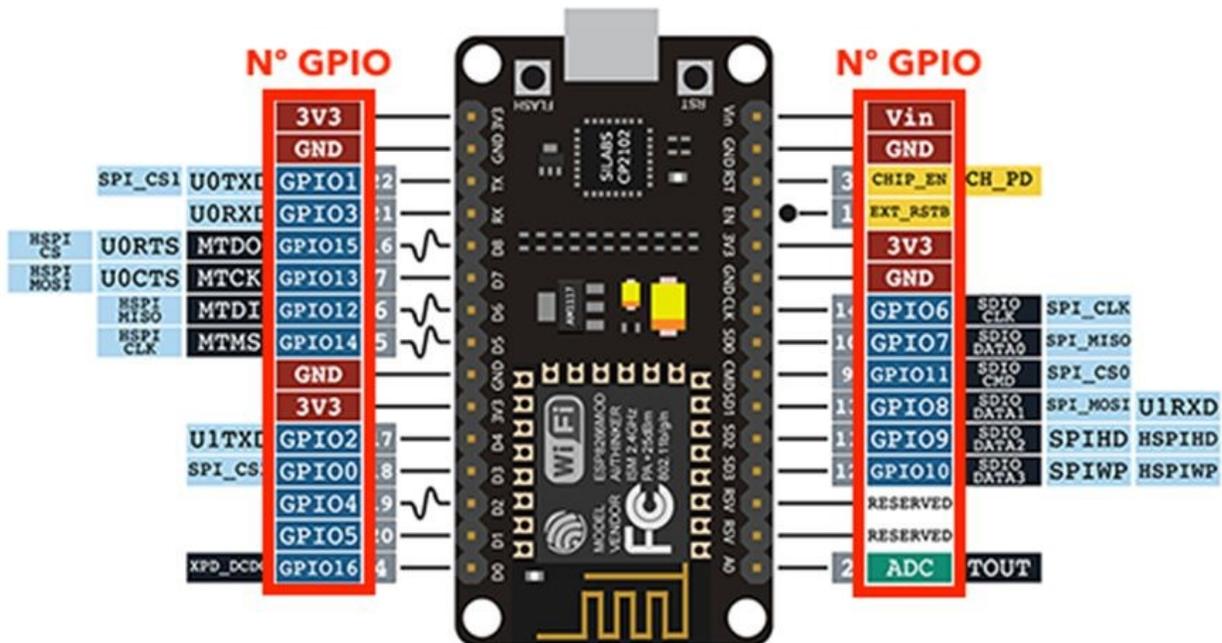


Figure 22 ESP 12 Pin out (MCU, 2019)

### 3.3.3 TCP – TRANSFER CONTROL PROTOCOL

It is an open internet protocol for data transfer over Wi-Fi. It is a versatile protocol which can be implemented in wide range of network types. This protocol can be applied to different levels and sizes of network architectures without interrupting the functionality or service. Each device is assigned with a unique IP address makes it easy to operate over complex network structure. The data transfer with this protocol uses acknowledgement signals to ensure data delivery. The speed of transfer is scalable exponentially to avoid collisions or data loss.

### 3.4 PC USER INTERFACE

The software which can be used to view the real time data on the user computer are Script communicator or MATLAB.

#### 3.4.1 SCRIPT COMMUNICATOR

Script Communicator is a scriptable console which used for communication through serial port. The most important desirable feature of this open source software is that it offers the user for custom signal rate (Baud rate), logging options including customizable console which can be scripted using java.

The transferred data is displayed in the terminal and logged in as html or text. The image of console of script communicator is shown below:

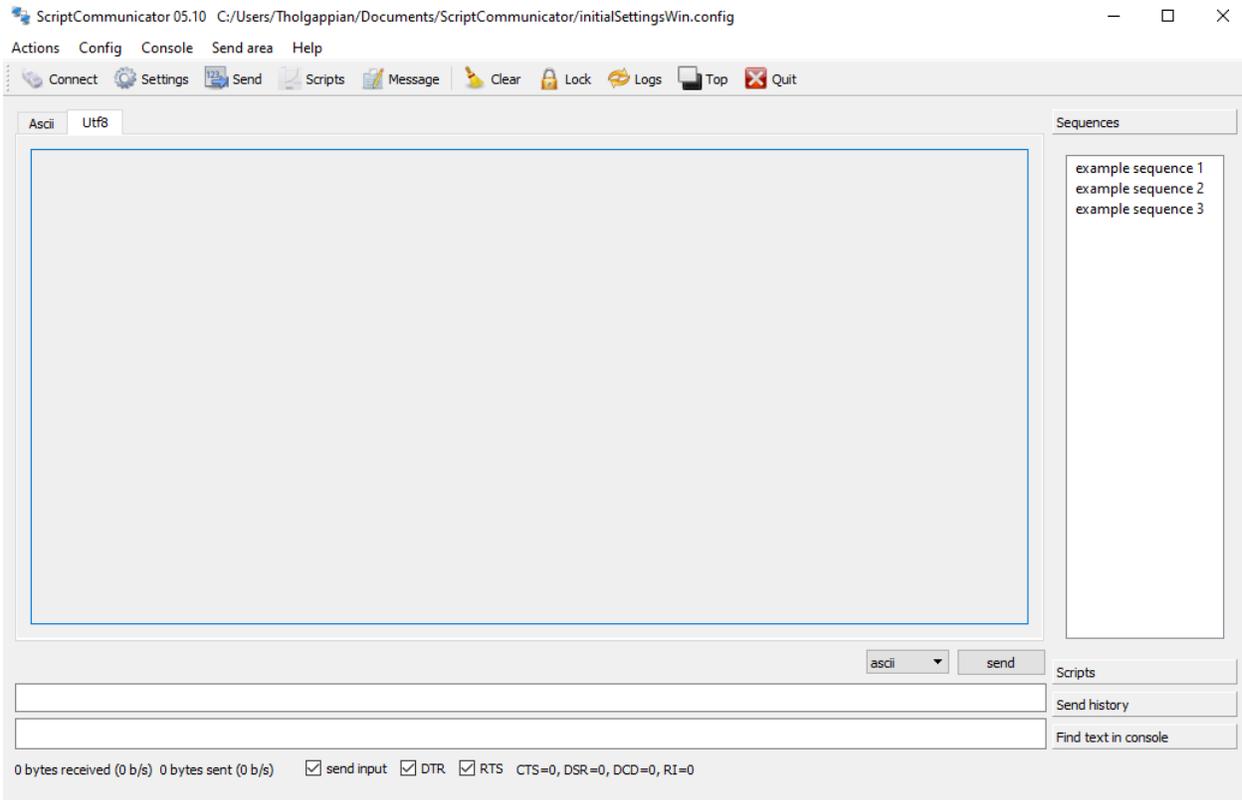


Figure 23 Data Logging Console

### 3.4.2 MATLAB

MATLAB can also be used to start serial connection and to read and transmit data over the connection. The major drawback faced during the usage of MATLAB during the process is data loss. Unlike TCP connection, the connection between the PC and the WI-FI shield is an efficiently managed. Because, serial port doesn't use acknowledgement flag, so the data sent over serial port can have the possibility to misread when the signal rate doesn't match. These signal mismatch can be due the fluctuating MATLAB compiler speed at minor levels. As soon as the compiler didn't get any data, it stops compilation and exits the process which is undesirable.

MATLAB/Simulink can be used for postprocessing once the sensor data are received and stored using Script communicator which prevents data loss.

## CHAPTER 4 PROGRAMMING THE SETUP

This chapter explains the procedures and strategies used in programming the setup. Once every desired electronics item was purchased, those things should be programmed to perform user desired function. Programming this setup requires basic programming knowledge in the domain of C /C++ from the user. The wirings found below used soldering wherever possible and jumper wires. The step by step procedures, wiring diagrams and program codes are reported under each section. This chapter completely explains the programming part from acquiring data from the sensor to storing or reading the sensor data at User PC. At the end of the chapter, control of steering motor by wire is explained clearly with useful diagrams.

### 4.1.1 ARDUINO IDE

Arduino IDE is an open source software used for writing and compiling the code for Arduino boards. It is an official Arduino software. Each Arduino microcontroller on the board that is actually programmed and accepts the information in the form of code. The main code, is called as sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment contains two things: Editor and Compiler.

Editor is used for writing the code for the board and Compiler is used for compiling and uploading the code to the connected Arduino Module. This environment supports both C and C++ languages.

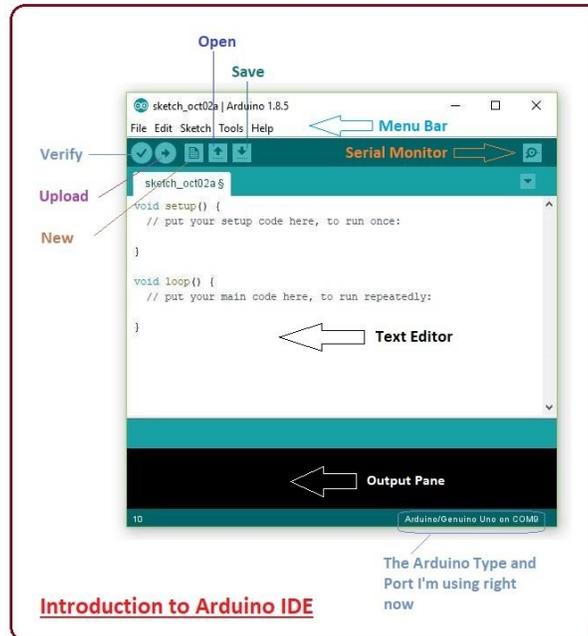


Figure 24 Arduino IDE (Arduino, 2019)

Keeping other startup things aside, each sketch (typical name used for Arduino codes) consists of two parts which are `setup ()` and `loop ()`.

The `setup ()` function is used to initiate all controller features, datatypes, etc. it is compiled only one time at the start of the Arduino. The `Loop ()` function is read in a loop continuously until the Arduino power is disabled. The frequency of `loop ()` iteration is based on the number of lines inside the function. Arduino takes particular time to read each line and the time increases with increase in solving complexity of the code.

## 4.2 ACQUIRING DATA FROM SENSOR

As said before the IMU MEMS sensor used here uses I2C protocol to communicate with micro controllers. The initial setup is to wire the Arduino with the sensor through i2c and reading the raw data like accelerations on three axes and angular velocity about three axes.

Any sensor data can be recorded by Arduino by calling the sensor hex address and start reading the data bits corresponding the registers. The process can be coded directly on Arduino through Arduino IDE or by just including IMU library. MPU 6050 sensor library for Arduino Due SAM Architecture is downloaded from git-hub open source platform and used since the library so far made available in the Arduino IDE only supports Atmel architecture but Arduino Due uses SAM architecture. The library compatible for SAM architecture is downloaded through git-hub.

```
void loop() {
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    Serial.print("a/g:\t");
```

```

Serial.print(ax); Serial.print("\t");
Serial.print(ay); Serial.print("\t");
Serial.print(az); Serial.print("\t");
Serial.print(gx); Serial.print("\t");
Serial.print(gy); Serial.print("\t");
Serial.println(gz);
}

```

The wiring diagram for reading MPU 6050 by Arduino is given below:

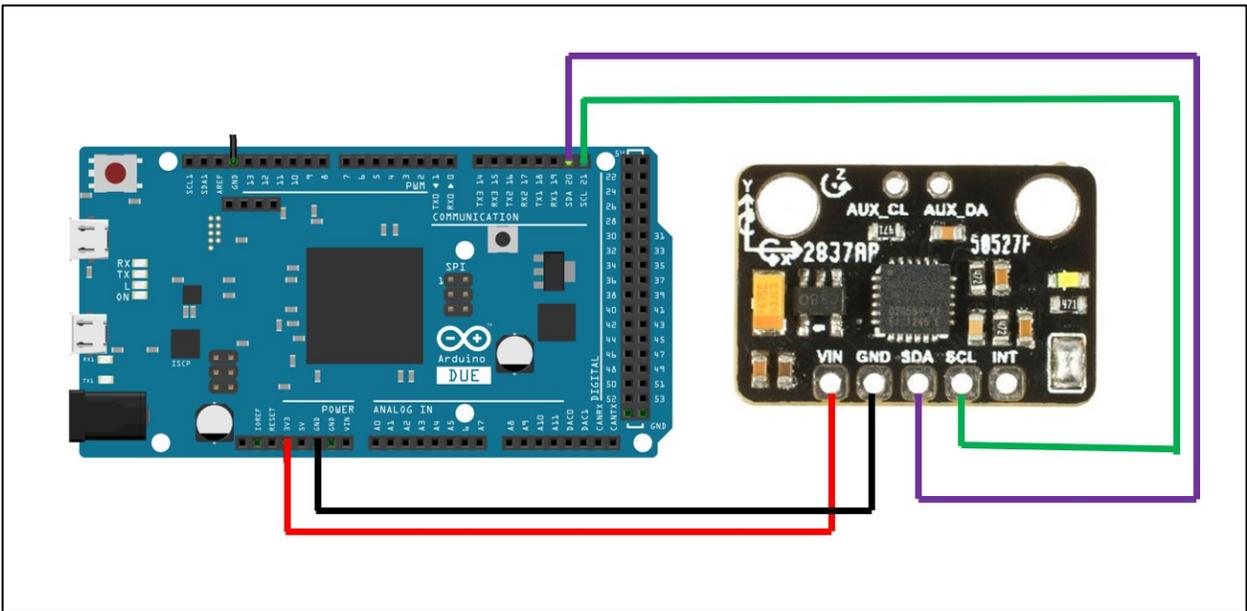


Figure 25 MPU 6050, Arduino Due Wiring diagram

Each library has a built-in raw data reading sample sketch form where further processes can be implemented. Raw data are the data which comes directly from the sensor and it is made user readable by converting the values using formulas advised by the sensor manufacturer.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2g	16384 LSB/g
1	±4g	8192 LSB/g
2	±8g	4096 LSB/g
3	±16g	2048 LSB/g

The concept here is that the raw data should be divided with appropriate LSB sensitivity value to convert in terms of g-values. The full-scale range can be changed to other ranges by changing the AFS\_SEL value in the library. The default sensitivity range is from -2g to +2g.

The same goes to gyro conversion as well.

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

The default full scale range is 2000 degrees per second.

MPU 6050 is one of the noisiest sensors and filtering is unavoidable. Accelerometers (and gyro meters as well) have significant noise, which usually does not zero-out via integration (It tends to be somewhat asymmetrical overall.)

The default sampling rate of the sensor is 100Hz which means the sensor emits 100 values per second for 200Hz internal clock. Although the sensor internal can work at 500Hz, in order to reduce duplicates 200Hz is advised by the manufacturer.

The values obtained from the sensor is averaged with 10 samples to eliminate higher order noise. Increasing the sample size can significantly affect the output speed. However, this is not going to affect much of our data since any potential filters can be applied during post processing. Another important approach used in reducing the noise is by controlling the decimal places.

\*\*\*\*\*

```
for (i = 1; i <=10; i += 1) // averaging i=10 samples
{
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // read measurements from device
  rax=ax;
  ray=ay;
  rgz=gz;
  avg_rgz=avg_rgz+rgz; // filtering gyro z and acc x and y
  avg_ray=avg_ray+ray;
  avg_rax=avg_rax+rax;
}
*****
```

So far from the initial setup the sensor gives values with six decimals which is high and non-desirable due to the noise. Hence reducing the decimal values to three which is far enough for post processing is done.

\*\*\*\*\*

```
//six decimals to three decimals conversion ax
    double q=avg_rax*1000;
    int w=q;
    q=w;
```

```

avg_rax=q/1000;
q=0;
//six decimals to three decimals conversion ay
q=avg_ray*1000;
w=q;
q=w;
avg_ray=q/1000;
q=0;
//six decimals to three decimals conversion gz
q=avg_rgz*1000;
w=q;
q=w;
avg_rgz=q/1000;
q=0;

```

\*\*\*\*\*

Time is an important parameter required for post processing. Arduino has a built-in clock which can be exploited for our use. The function used to retrieve the Arduino system time which runs in milliseconds is `millis()`.

`unsigned long t=millis();` (this code leaves time to the integer “t”)

### 4.3 DATA TRANSMISSION

Once the data is read from the sensor the next step is to transfer it to user PC via Wi-Fi. The Wi-Fi shield used here is ESP 12 Node MCU 1.0. Node MCU is itself a microcontroller which is able to do small tasks. Stressing the transmission device more can leave us data loss hence no additional program or task is done by the Node MCU here for immediate data transfer.

The communication used between Arduino due and Node MCU is UART serial interface. Other possible communication is I2C. Although Arduino due has two I2C ports, the second I2C ports such as SCL1 and SDA1 requires an additional pullup resistor for its functioning without damaging the board. As implied by the Arduino datasheet, the board is functional with 3.3V input. Inputting more than that can significantly damage the board. Multiplexing SDA and SCL is also one of the ways, but it requires an I2C multiplexer which is of extra cost.

The possibility of data loss in UART communication is high if the compiler speeds of the Arduino and ESP 12 is not matching. The data transfer in UART is asynchronous which means no clock data is transferred. So, any glitches while functioning can cause data loss. This is also one of the prime reasons to avoid loading extra programming task from the node MCU side.

The wiring diagram between Node MCU and Arduino Due is given below:

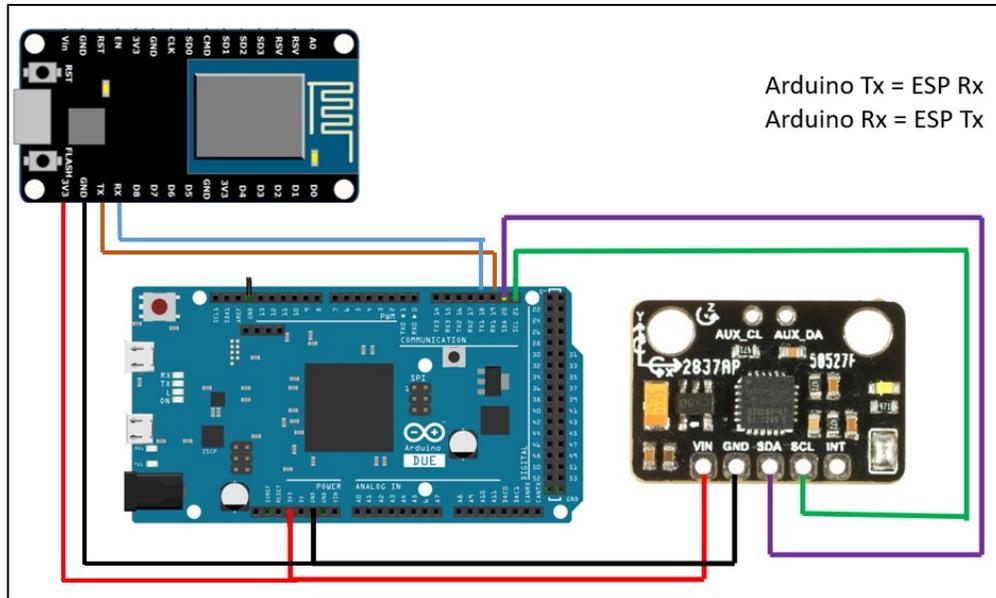


Figure 26 Arduino Due, ESP 12, MPU 6050 Wiring Diagram

\*\*\*\*\*

```
// From Arduino
snprintf(m,60,"<%lu,%f,%f,%f>",t,avg_rax,avg_ray,avg_rgz); (printing Ax,Ay and Gz)
Serial.println(m);
Serial1.println(m); (Prints the string in UART communication channel)
delay(1); (One millisecond delay to prevent overlap)
```

\*\*\*\*\*

It is possible to send values one by one, instead of that “snprintf” syntax combines all the numbers into a single string with preallotted data size. This strategy can eliminate latency and data loss during asynchronous transmission.

“<,>” used here as the boundaries of the string useful in eliminating misread errors.

Example :

```
<4027,0.000000,0.000000,0.000000|>
<time, Accel_x, Accel_y, Gyro_z|>
```

```
String s=Serial.readStringUntil('>'); (Reads the string from Arduino until ">")
Serial.print(s); (Prints the string)
```

### 4.3.1 SETTING UP Wi-Fi: TRANSMITTER

ESP 12E is specifically designed with enhanced features for Wi-Fi data transfer. The below program enables the device to transfer the data read from the Arduino to the receiver using software enabled (Soft-AP) Wi-Fi. The Wi-Fi is WPA encrypted. This code must be uploaded to the ESP 12 connected with Arduino.

```
*****  
  
#include <ESP8266WiFi.h>      ( Header to call ESP 12E library)  
  
WiFiServer server(80);        }  
IPAddress IP(192,168,4,15);   } Setting up Wi-Fi configuration with desired IP and Gateway  
IPAddress mask = (255, 255, 255, 0);  
  
void setup()  
{  
  Serial.begin(250000);      (Open serial communications and wait for port to open)  
  while (!Serial) {          ( waits until the serial port connects.)  
    ; }  
  
  WiFi.mode(WIFI_AP);  
  WiFi.softAP("Traxxas_AP", "Wemos_comm"); }  
  WiFi.softAPConfig(IP, IP, mask);         } Setting up WI-FI Access point with SSID:Traxxas_AP and  
  server.begin();                          } Password: Wemos_comm.  
  Serial.println();                        } *softAP – Software enabled AP  
  Serial.println("Server started.\n");  
  Serial.print("IP: \n");  
  Serial.println(WiFi.softAPIP());  
  Serial.print("MAC:\n");  
  Serial.println(WiFi.softAPmacAddress()); } Prints the soft Wi-Fi details  
  delay(10);  
}  
  
void loop() {  
  WiFiClient client = server.available(); } Starts the data transfer only and only if any device receives it  
  if (Serial.available()) {  
    String s=Serial.readStringUntil('>'); } Reads the string until the delimiter defined (">")  
    Serial.print(s);  
    client.println(s);  
    client.flush(); } Clears the present data buffer for next transfer  
  }  
}
```

### 4.3.2 SETTING UP Wi-Fi: RECEIVER

The below program enables the device to read the data from the ESP 12 connected with Arduino through software enabled (Soft-AP) Wi-Fi with defined WPA login credentials. This code must be uploaded to the ESP 12 connected with User PC.

\*\*\*\*\*

```
#include <ESP8266Wi-Fi.h>
```

```
char ssid[] = "Traxxas_AP";  
char pass[] = "Wemos_comm";  
IPAddress server(192,168,4,15);  
WiFiClient client;
```

Wi-Fi credentials declaration

```
void setup() {  
  Serial.begin(250000);  
  
  WiFi.mode(WI-FI_STA);  
  WiFi.begin(ssid, pass);  
  Serial.println();  
  Serial.println("Connection to the AP");  
  while (WiFi.status() != WL_CONNECTED) {  
    Serial.print(".");  
    delay(5);  
  }
```

Starting connection with above credentials

```
  Serial.println();  
  Serial.println("Connected");  
  Serial.print("LocalIP:"); Serial.println(WiFi.localIP());  
  Serial.println("MAC:" + WiFi.macAddress());  
  Serial.print("Gateway:"); Serial.println(WiFi.gatewayIP());  
  Serial.print("AP MAC:"); Serial.println(WiFi.BSSIDstr());
```

Printing the soft Wi-Fi details for the

```
}  
void loop()  
{  
  client.connect(server, 80);  
  String r=client.readStringUntil("|");  
  Serial.print(r);  
  
  client.flush();  
  client.stop();  
}
```

Reads the string until the delimiter "|" and prints it to the serial port

Clears the present data buffer

\*\*\*\*\*

So finally, the data received from the user's monitor will be like the following format:

Example:

`<4027,0.000000,0.000000,0.000000`

`<time, Accel_x, Accel_y, Gyro_z`

- Without the delimiters

## 4.4 STEERING MOTOR CONTROL

As previously said, the servo motor is able to work with Pulse Width Modulation (PWM). So, it may seem to be easy to connect with the Arduino, as Arduino has 54 digital pins where PWM is easy to perform.

But considering the power rating of the servo motor which works at 14V and high current, makes it impossible to feed via Arduino as max current tolerance is 125 mA. Drawing more current from the Arduino PWM digital pin can fry the board.

So, there are two ways to make it function

1. Motor driver
2. External power source

In the first method, using a motor driver requires additional components.

But in the second method, the external power source can be the R/C car's power source, which is the batteries. It is possible to feed power from the battery and feed control input from the microcontroller. Thus, making it simple to control the steering servo.

The wiring diagram for steering motor control is given below:

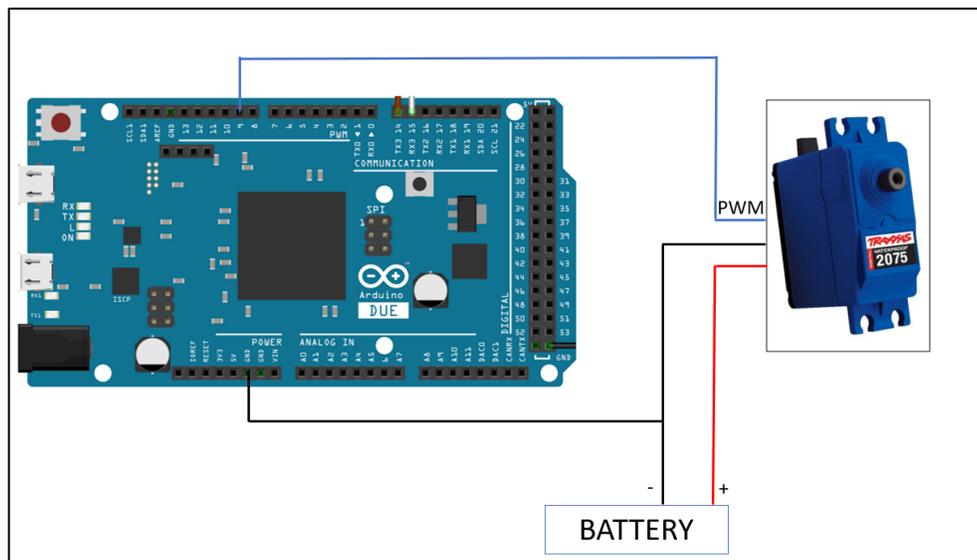


Figure 27 Steering servo, Arduino Due Wiring diagram

\*\*\*\*\*

```
//servo library

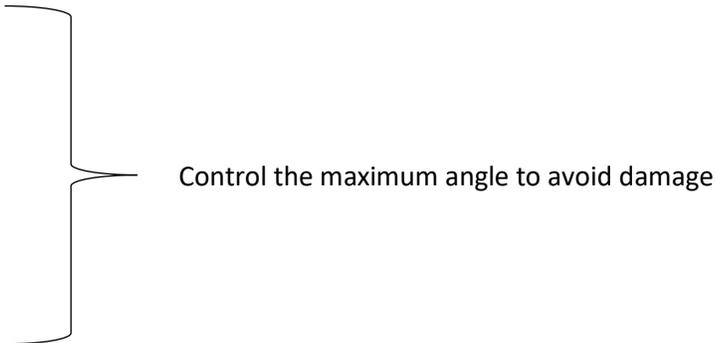
#include <Servo.h>
Servo myservo;           // Create servo object to control a servo
int steer_offset=94;

void setup()
{
myservo.attach(9);      //Attach PWM 9 Pin to servo motor
}
void loop()
{
int deg_output=tiredeg;
if(tiredeg>27)
{
deg_output=27;
}
if(tiredeg<-27)
{
deg_output=-27;
}

deg_output=deg_output*84/54;    // Mapping the values to motor scale

pos=steer_offset+ deg_output;

myservo.write(pos);           // Writing the position to the servo motor
}
```



\*\*\*\*\*

The maximum tire angle on the setup is 27 degrees. So, it is mandatory to restrict angles more than that to avoid damages to the servo motor. Then the tire degree value is mapped or scaled to servo motor position.

There is a steering offset of 94 degrees, which implies, the tire angle is zero for servo position of 94 degrees. It is possible to timely control the steering by calling separate timer functions.

## CHAPTER 5 SETUP VALIDATION

The setup should facilitate the future researchers and engineers to perform indoor testing (over a test bed as well as outdoor testing on custom designed circuits. In order to confidently test vehicles in future, the primary step is to validate the setup, including measurements.

For that, the R/C car is subjected to run on a preconfigured longitudinal velocity and steering input of user's interest and the response is measured through accelerometers and gyros, and logged for further analysis. Since the aim is to study the planar dynamics, and by referring to the linear planar dynamics equations given in (Jazar, 2017),

$$\dot{v}_x = \frac{F_x}{m} + r v_y$$

$$\dot{v}_y = \frac{1}{m v_x} (-a C_{af} + b C_{ar}) r - \frac{1}{m v_x} (C_{af} + C_{ar}) v_y + \frac{1}{m} C_{af} \delta_f + \frac{1}{m} C_{ar} \delta_r - r v_x$$

$$\dot{r} = \frac{1}{I_z v_x} (-a^2 C_{af} - b^2 C_{ar}) r - \frac{1}{I_z v_x} (a C_{af} - b C_{ar}) v_y + \frac{1}{I_z} a C_{af} \delta_f - \frac{1}{I_z} b C_{ar} \delta_r$$

The equations can be converted to matrix form as below,

$$\begin{bmatrix} \dot{v}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{C_{af} + C_{ar}}{m v_x} & \frac{-a C_{af} + b C_{ar}}{m v_x} - v_x \\ -\frac{a C_{af} - b C_{ar}}{I_z v_x} & -\frac{a^2 C_{af} + b^2 C_{ar}}{I_z v_x} \end{bmatrix} \begin{bmatrix} v_y \\ r \end{bmatrix} + \begin{bmatrix} \frac{1}{m} C_{af} & \frac{1}{m} C_{ar} \\ \frac{1}{I_z} a C_{af} & -\frac{1}{I_z} b C_{ar} \end{bmatrix} \begin{bmatrix} \delta_f \\ \delta_r \end{bmatrix}$$

Where  $v_x$  is longitudinal velocity,  $v_y$  is lateral velocity,  $r$  is yaw rate,  $a$  is the distance between front axle and Center of gravity location,  $b$  is distance between rear axle and center of gravity location,  $C_{af}$  is front

axle cornering stiffness,  $C_{ar}$  is rear axle cornering stiffness,  $I_z$  is mass moment of inertia and  $\delta_f$  is steering input at front wheels,  $\delta_r$  is steering input at rear wheels and  $m$  is mass. To use the above equations, certain parameters and variables are required to be calculated or estimated or measured.

The R/C car is equipped with, 3-axis acceleration and steering angle. But the question arises, how the variables like cornering stiffness( $C\alpha$ ), vehicle slip( $\beta$ ), longitudinal velocity ( $v_x$ ) can be estimated?

### 5.1 YAW MOMENT OF INERTIA ESTIMATION

The measurement of yaw moment of inertia will be little challenging and little of accuracy must be sacrificed since the vehicle system is asymmetric in Y and Z axes and hence bifilar pendulum may be used to derive the value. The R/C car is subjected to yaw oscillations and frequency is measured to calculate yaw moment of inertia value using free vibration dynamics equations.

The equation to estimate the yaw moment of inertia is given by (Genta & Delprete, 1994),

$$I_z = \frac{M g b^2 T^2}{4\pi^2 h}$$

Where  $M$  is the mass of the object of interest,  $g$  is the acceleration due to gravity,  $D$  is the distance between the parallel strings( $D=2b$ ),  $T$  is the time taken for one oscillation and  $h$  is the length of the string.

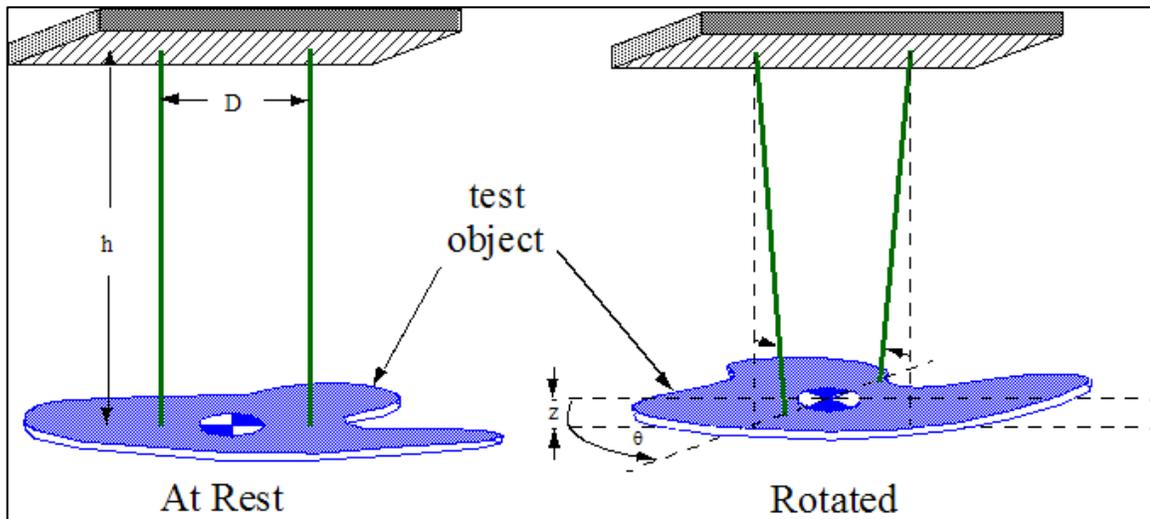


Figure 28 Mass moment of inertia about Z axis – Bifilar pendulum technique

To ensure that the Inertia measurement setup has better accuracy, the setup is subjected to linear oscillations and frequency is noted and substituted in the following equation to check whether the values generate acceptable value of acceleration due to gravity.

$$\text{Acceleration due to gravity } 'g' = \omega^2 h = \frac{4\pi^2}{T^2} h$$

Where  $\omega = 2\pi f$ , T is average time taken for one linear oscillation

Vehicle inertia measuring machine (VIMM) (Schiller, 2019) can measure nearly 10 inertia parameters such as vehicle mass, center of gravity, moments of inertia  $I_x$   $I_y$   $I_z$  and deviation moments  $I_{xy}$   $I_{yz}$   $I_{zx}$  for full sized vehicle. But to study planar dynamics (only  $I_z$  required), bifilar pendulum is sufficient and cost effective.



Figure 29 Bifilar Pendulum assembled

### Calculation of MOI

Given data

$$\text{Weight} = mg = 3.9 + (\text{batteries} * 2) = 4.52 \text{ Kg}$$

$$h = 83 \text{ cm} = 0.83 \text{ m}$$

$$D = 46 \text{ cm}$$

$$b = D/2 = 23 \text{ cm} = 0.23 \text{ m}$$

Verification of the setup

The setup is initially subject to linear displacement as in simple pendulum and the time is noted for cross verification of setup accuracy.

Iterations	Time for 10 oscillations
1	18.31
2	18.35
3	18.34

Hence the average time for one oscillation would be 1.83 seconds

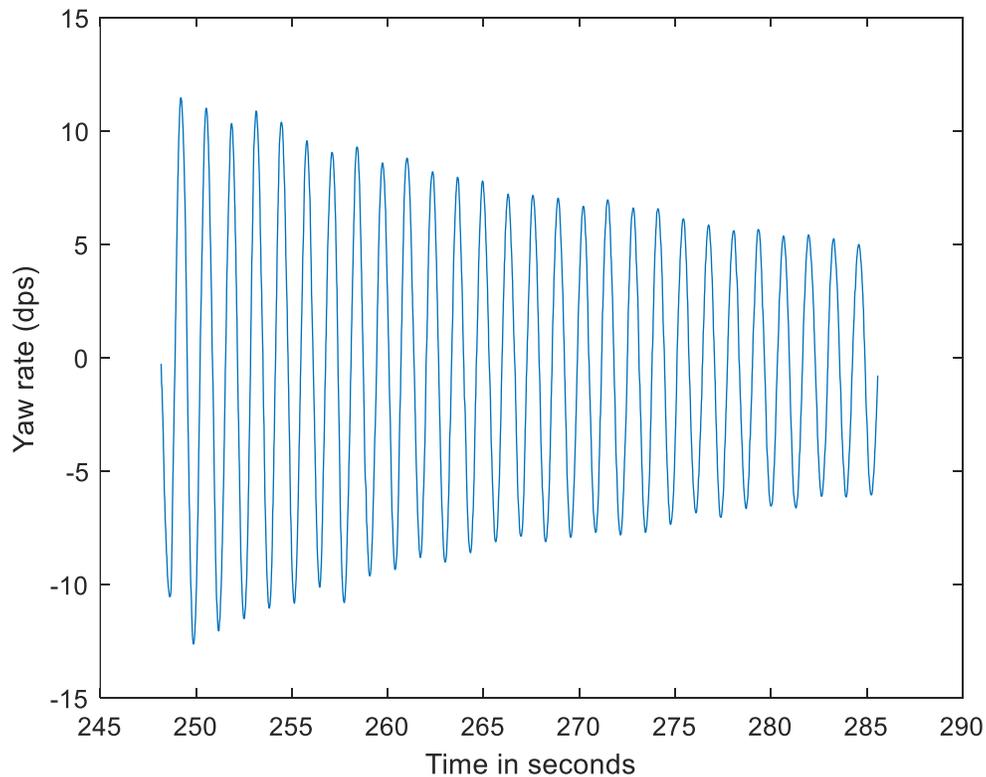
Acceleration due to gravity  $g = \omega^2 h = \frac{4\pi^2}{T^2} h$

$$g = \frac{4 * 3.14 * 3.14 * 0.83}{1.83 * 1.83} = 9.79 \text{ m/s}^2 \text{ which is close to } 9.81 \text{ m/s}^2.$$

So, the setup is valid for further experimentation.

Bifilar pendulum concept

The setup is then subjected to minimum angular displacement about Z axis and the time for an oscillation is noted.



Hence from the graph time for one oscillation would be 1.4 seconds. Substituting in the below equation to calculate mass moment of inertia,

$$I_z = \frac{M g b^2 T^2}{4\pi^2 h}$$
$$I_z = \frac{4.52 * 9.81 * 0.23 * 0.23 * 1.4 * 1.4}{4 * 3.14 * 3.14 * 0.83}$$

$$I_z = 0.14 \text{ Kgm}^2$$

The calculated value is a close approximation for moment of inertia about Z axis.

## 5.2 CORNERING STIFFNESS ESTIMATION

Estimation of cornering stiffness is done by various methods like static test bed method by Lapapong et al. (2009), beta less method used by *Sierra et al. (2006)* or separate tire testing on a testbed equipped with 6-axis force transducers by *Polley and Alleyne (2004)*.

**The beta less method** uses intermediate variables to deduce the ratios of stiffnesses and weighting with least square estimation. Beta less method makes use of yaw rate ( $r$ ), lateral acceleration ( $a_y$ ) and vehicle yaw moment of inertia ( $I_{zz}$ ) to form an equation independent of side slip angles.

$$\begin{bmatrix} mL a_y & L \left( \delta - \frac{Lr}{u} \right) \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = I_{zz} \dot{r} + m b a_y,$$

where

$$X_1 \equiv \frac{C_{\alpha f}}{C_{\alpha f} + C_{\alpha r}} \quad \text{and} \quad X_2 \equiv \frac{C_{\alpha f} C_{\alpha r}}{C_{\alpha f} + C_{\alpha r}}.$$

While the above methods seem feasible but the considering the conditions of testing in acquiring the parameter values makes it difficult to take forward. For example, the beta less method gives worse approximation when the yaw acceleration is zero and adding weights to the values leaves clueless in deducing the values.

Hence an alternative approach is used to deduce the cornering stiffness values.

### CALCULATION OF CORNERING STIFFNESS

To calculate cornering stiffness of the tires, certain measurements are needed such as vehicle slip, yaw rate, actual vehicle lateral and longitudinal velocity.

It is given that

$$A_y = \dot{V}_y + rV_x; \text{ ----- (1)}$$

$$A_x = \dot{V}_x - rV_y; \text{ -----(2)}$$

Under test conditions, the R/C car is subjected to steady state condition at constant steering angle of 20 degrees. The vehicle is subjected to various speed test at 5kmph, 10 kmph, 15kmph and 20 kmph. The below graphs corresponds to 20kmph test.

The acquired data is plotted below;

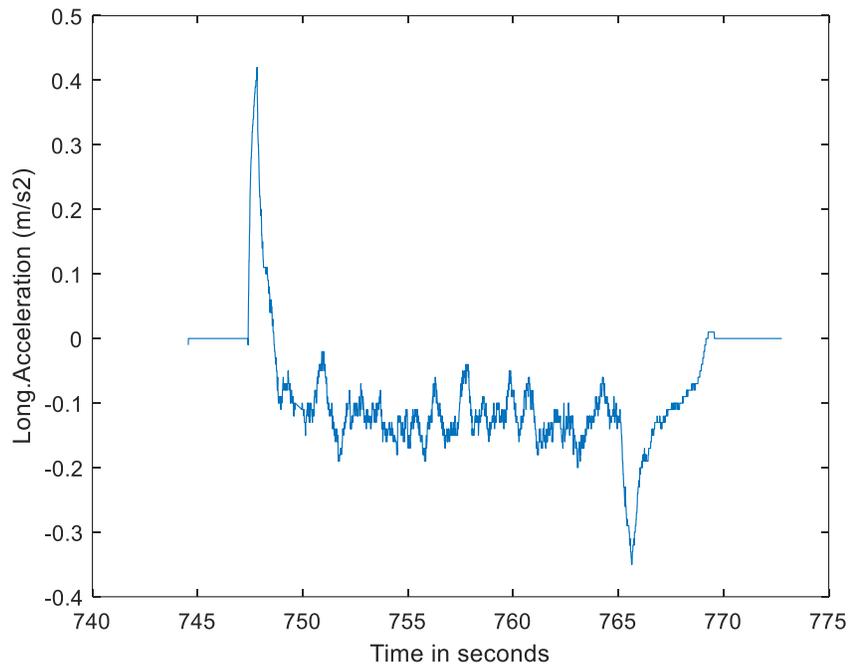


Figure 30 Longitudinal Acceleration Vs Time

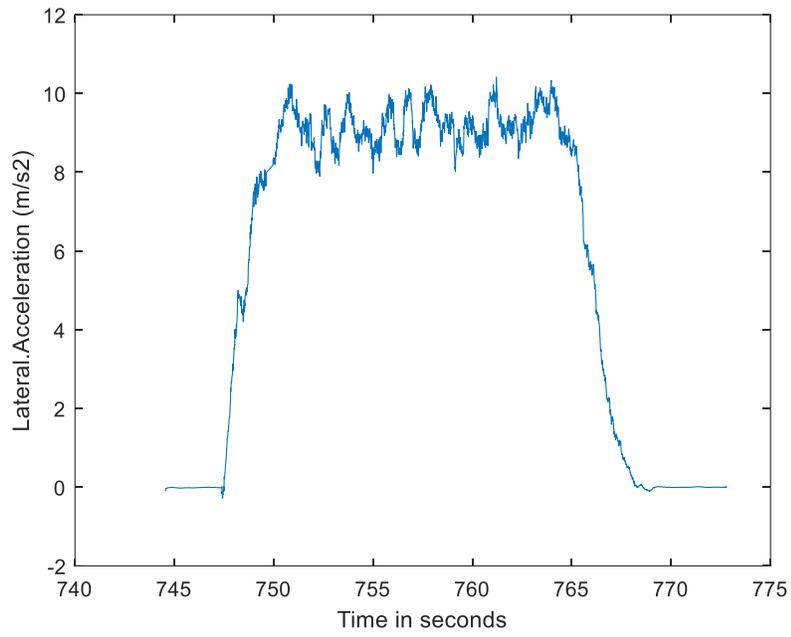


Figure 32 Lateral Acceleration Vs Time

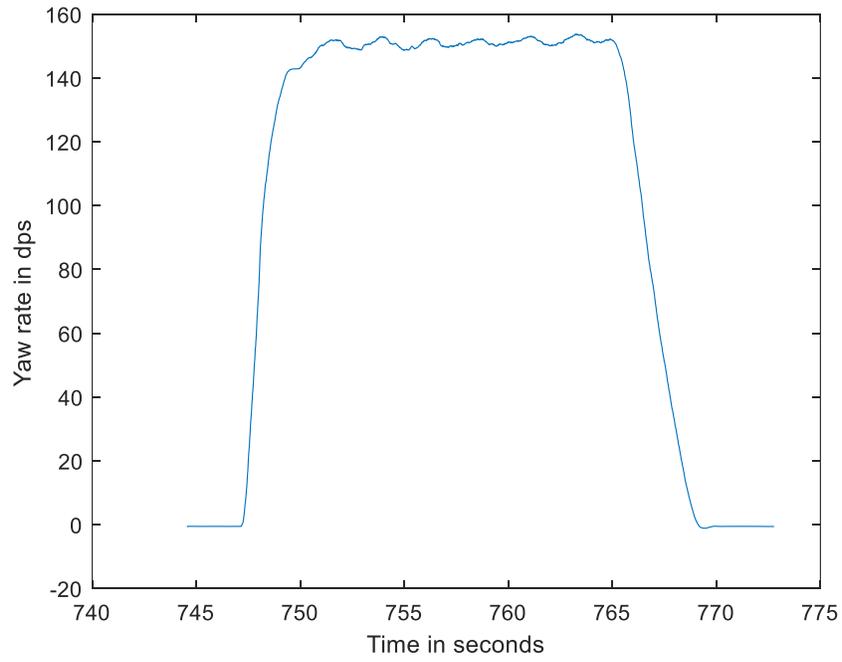


Figure 31 Yaw rate Vs Time

The figure shows the acceleration and velocities under steady state high speed cornering condition.

At steady state condition, from equation (1),

$$\dot{V}_y = 0$$

So, lateral acceleration figure plotted is equated as below,  
Y acceleration at steady state,

$$A_{y_{ss}} = r \cdot V_x \quad (\text{From equation (1)}) \quad \text{----- (3)}$$

$$A_{y_{ss}} = 9.2 \quad (\text{from the plot})$$

Yaw rate  $r = 151 \text{ dps}$  (from the plot figure) = 2.64 radians per second

Substituting in above equation (3),

$$A_{y_{ss}} = r \cdot V_x$$

$$9.2 = 2.64 * V_x$$

$V_x = 3.5 \text{ m/s}$ , which is the vehicle's actual longitudinal velocity.

From the plot,

$$-rV_y = -0.1$$

Substituting  $r$  value, gives  $V_y = 0.04 \text{ m/s}$ .

Calculating  $\beta$ ,

$$\begin{aligned} \text{Vehicle slip } (\beta) &= \frac{V_y}{V_x} = \frac{0.04}{3.5} \\ &= 0.01 \text{ radians} \\ &= 0.65 \text{ degrees} \end{aligned}$$

Calculating slips,

$$\begin{aligned} \alpha_1 &= \beta + \frac{ar}{V_x} - \delta \\ \alpha_1 &= 0.01 + \frac{0.224 * 2.64}{3.5} - 0.35 \\ \alpha_1 &= -0.17 \text{ radians} \\ \alpha_1 &\approx -10.9 \text{ degrees} \end{aligned}$$

Similarly

$$\begin{aligned} \alpha_2 &= \beta - \frac{br}{V_x} \\ \alpha_2 &= 0.01 - \frac{0.18 * 2.64}{3.5} \\ \alpha_2 &= -0.13 \text{ radians} \\ \alpha_2 &\approx -8.6 \text{ degrees} \end{aligned}$$

Vehicle lateral force is given by

$$\begin{aligned}F_y &= m * a_y \\ &= 4.52 * 0.9 * 9.81 \\ &= 39.8 \text{ N}\end{aligned}$$

Side force at front axle

$$\begin{aligned}F_{yf} &= F_y * \frac{b}{L} \\ F_{yf} &= 39.8 * \frac{0.18}{0.404} \\ F_{yf} &= 17.76 \text{ N}\end{aligned}$$

The component along tires y axis is given by

$$\begin{aligned}F_{yf}^* &= \frac{F_{yf}}{\cos(20)} \quad * 20 \text{ degrees steering} \\ F_{yf}^* &= 18.89 \text{ N}\end{aligned}$$

Side force at rear axle

$$\begin{aligned}F_{yr} &= F_y * \frac{a}{L} \\ F_{yr} &= 39.8 * \frac{0.224}{0.404} \\ F_{yr} &= 22.02 \text{ N}\end{aligned}$$

$$\text{Cornering stiffness Front axle } C_f = \frac{F_{yf}}{\alpha_1} = \frac{18.89}{-0.19} = 99 \text{ Nrad}^{-1}$$

$$\text{Cornering stiffness Rear axle } C_r = \frac{F_{yf}}{\alpha_2} = \frac{22.02}{-0.146} = 150 \text{ Nrad}^{-1}$$

The same steps were followed with other speed ranges and the results were around the same values. Rear stiffness is greater than front due to the fact that COG is more towards rear and width of rear tire is 1.5 times bigger than front.

Since the radius of curvature is increased, the vehicle is an understeering vehicle.

### 5.3 VALIDATING WITH BICYCLE MODEL

Let's verify whether the measured radius of curvature  $\approx 1.27$  m is replicated by the above values on substituting in the equations of bicycle model.

#### 5.3.1 CURVATURE GAIN

Curvature gain can be given by

$$\frac{1}{R\delta} = \frac{1}{L} * \frac{1}{1 + \frac{mV^2}{L^2} \left( \frac{b}{C_f} - \frac{a}{C_r} \right)}$$

On substituting the values acquired by the experiment such as  $C_f$ ,  $C_r$ ,  $L$  m,  $V$  and  $\delta$ ,

$R$  value can be given by

$$\frac{1}{R*0.35} = \frac{1}{0.404} * \frac{1}{1 + \frac{4.516*3.49^2}{0.404^2} \left( \frac{0.18}{99} - \frac{0.224}{150} \right)}$$

$R = 1.274\text{m} \approx$  experimented  $R$   $1.27\text{m}$

#### 5.3.2 UNDERSTEERING COEFFICIENT

$$\begin{aligned} K_{us} &= \frac{m*g}{L} * \left( \frac{b}{C_f} - \frac{a}{C_r} \right) \\ &= \frac{4.516*9.81}{0.404} * \left( \frac{0.18}{99} - \frac{0.224}{150} \right) \end{aligned}$$

$$= 109.65 * 0.0003 = 0.036 \quad K_{us} > 0 \text{ (Understeering)}$$

#### 5.4.3 RESPONSES- STEADY STATE

The derived values are substituted in various response equations such as beta response, lateral velocity response, curvature gain response, yaw rate response and its values are verified against the sensor data.

The equations for each response in given below:

Curvature gain  $S_\kappa = \frac{lC_{\alpha f}C_{\alpha r}}{m(C_{\alpha r}a_2 - C_{\alpha f}a_1)v_x^2 + l^2C_{\alpha f}C_{\alpha r}}$

Vehicle slip  $S_\beta = \frac{a_2lC_{\alpha f}C_{\alpha r} - ma_1v_x^2C_{\alpha f}}{m(C_{\alpha r}a_2 - C_{\alpha f}a_1)v_x^2 + l^2C_{\alpha f}C_{\alpha r}}$

Yaw rate  $S_r = \frac{v_x l C_{\alpha f} C_{\alpha r}}{m(C_{\alpha r}a_2 - C_{\alpha f}a_1)v_x^2 + l^2C_{\alpha f}C_{\alpha r}}$

Lateral velocity  $S_y = \frac{a_2lC_{\alpha f}C_{\alpha r} - ma_1v_x^2C_{\alpha f}}{m(C_{\alpha r}a_2 - C_{\alpha f}a_1)v_x^2 + l^2C_{\alpha f}C_{\alpha r}}v_x$

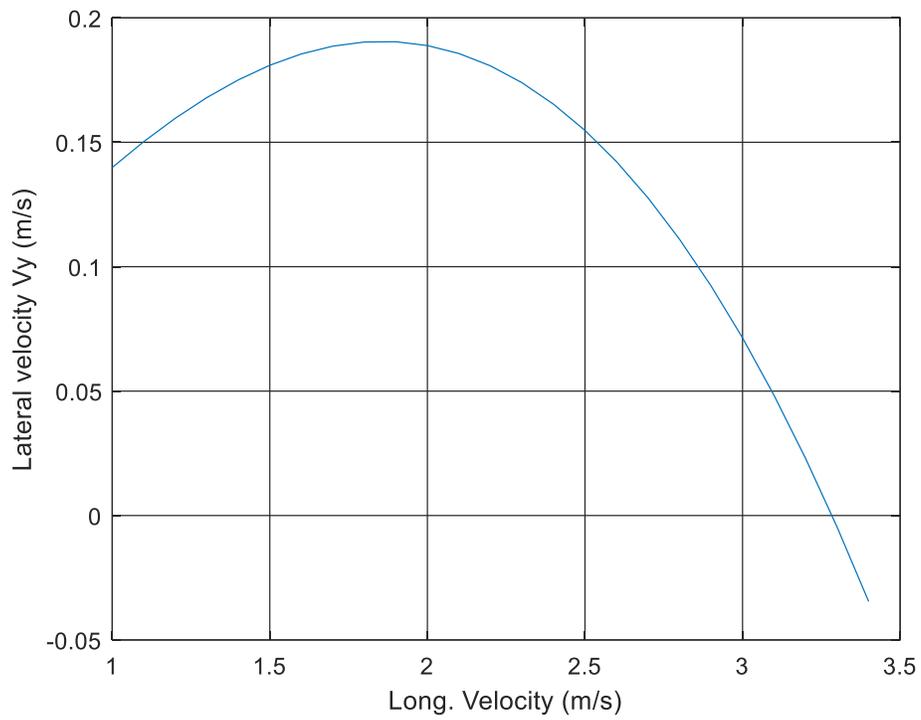


Figure 34 Lateral Velocity response vs Longitudinal velocity

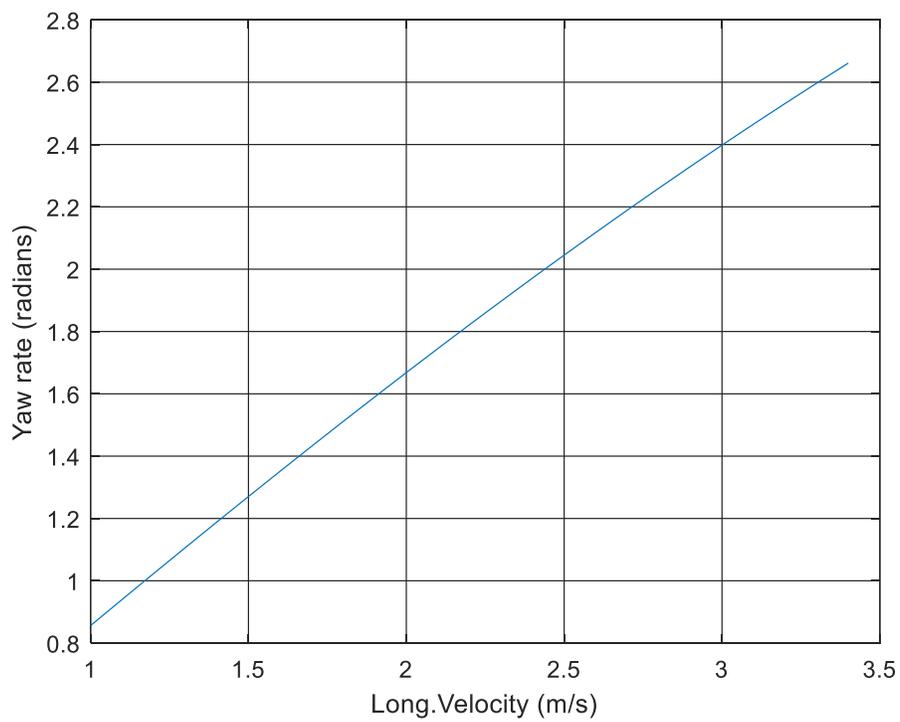


Figure 33 Yaw rate response vs Longitudinal Velocity

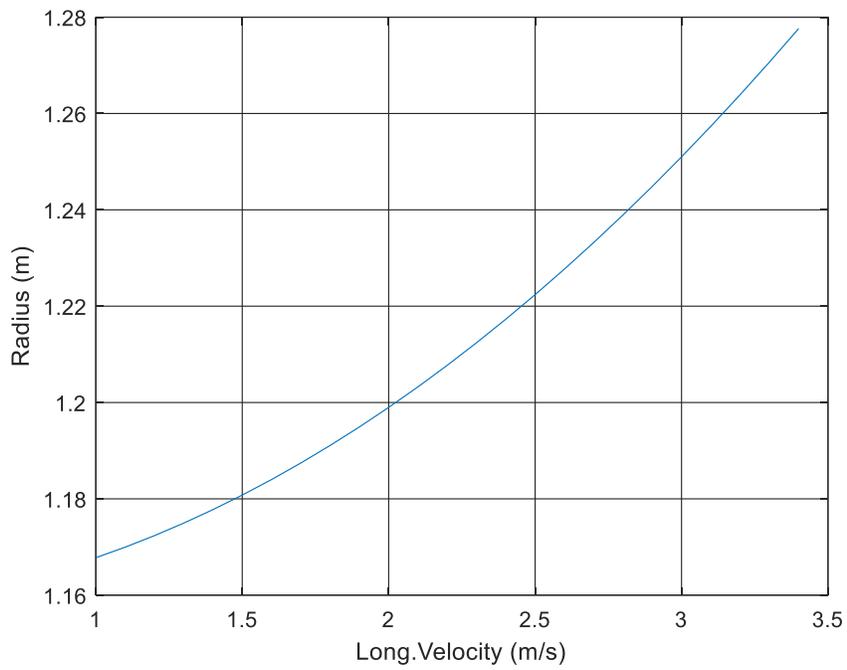


Figure 36 Radius of curvature vs Longitudinal velocity

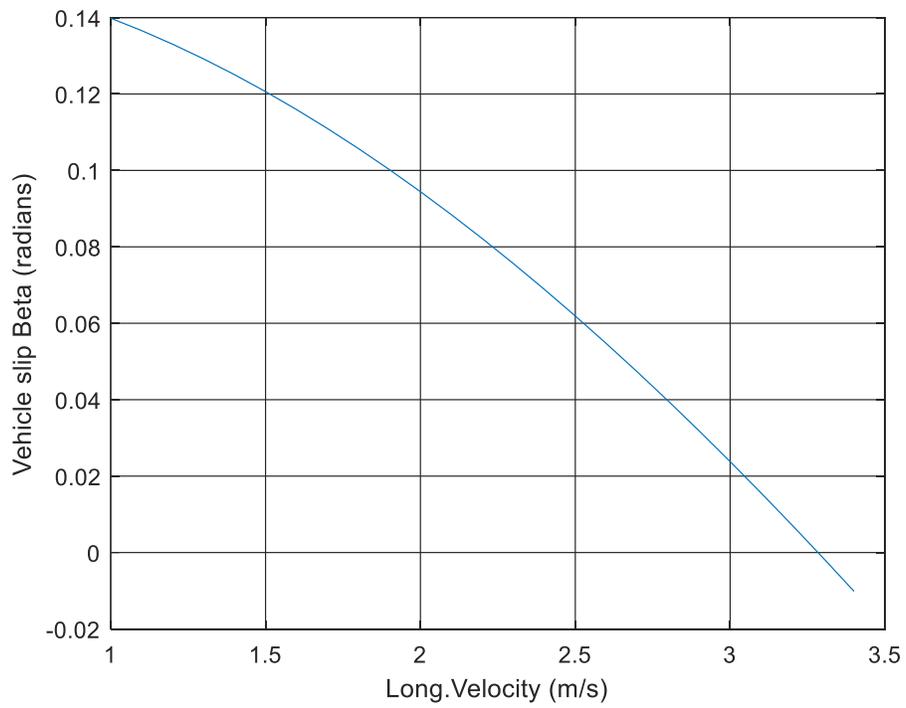


Figure 35 Vehicle slip vs Longitudinal velocity

From the above plots, the values yielded at the longitudinal velocity 3.4 m/s is given as follows:

Vehicle slip  $\beta = 0.01$  radians

Lateral velocity  $V_y = -0.03$  m/s

Yaw rate ( $r$ ) = 2.66 radians per second

Curvature gain  $R = 1.28$  m

The above values are fairly close enough to the derived values. Hence it is evident that the setup absolutely follows and delivers theoretical bicycle model in steady-state. We conclude the hardware and measurement setup is properly implemented and the data acquisition is reliable.

#### 5.4.4 OTHER RESPONSES

At transient state, State space vehicle dynamics model can be expressed as

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

$$A = \begin{bmatrix} -\frac{C_f + C_r}{mU} & \frac{bC_r - aC_f}{mU} - U \\ \frac{bC_r - aC_f}{I_{zz}U} & -\frac{a^2C_f + b^2C_r}{I_{zz}U} \end{bmatrix}$$

$$B = \begin{bmatrix} C_f & aC_f \\ m & I_{zz} \end{bmatrix}^T$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; D = 0; u = \delta;$$

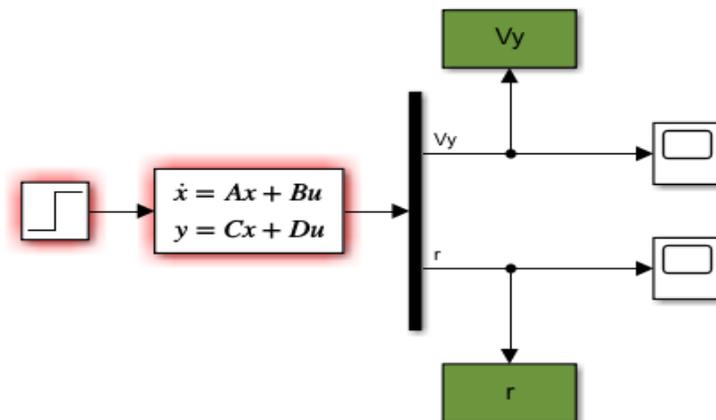


Figure 37 Simulink Model

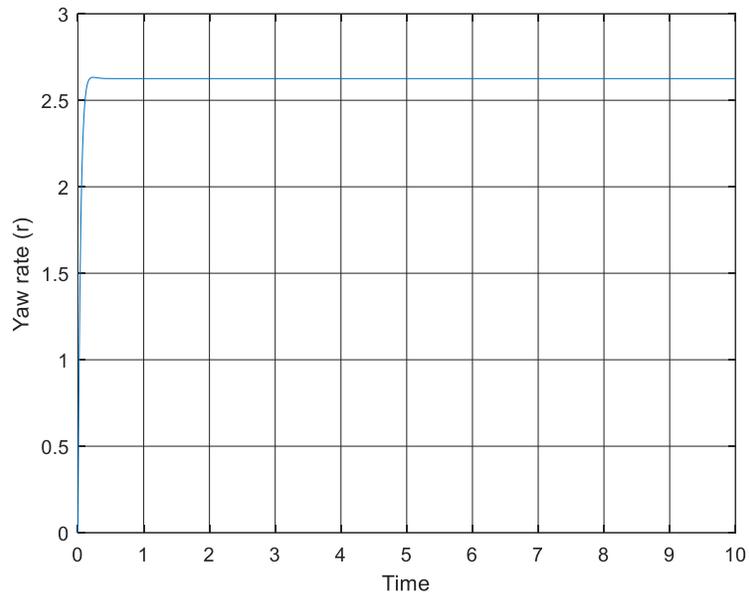


Figure 39 Yaw rate response Vs Time

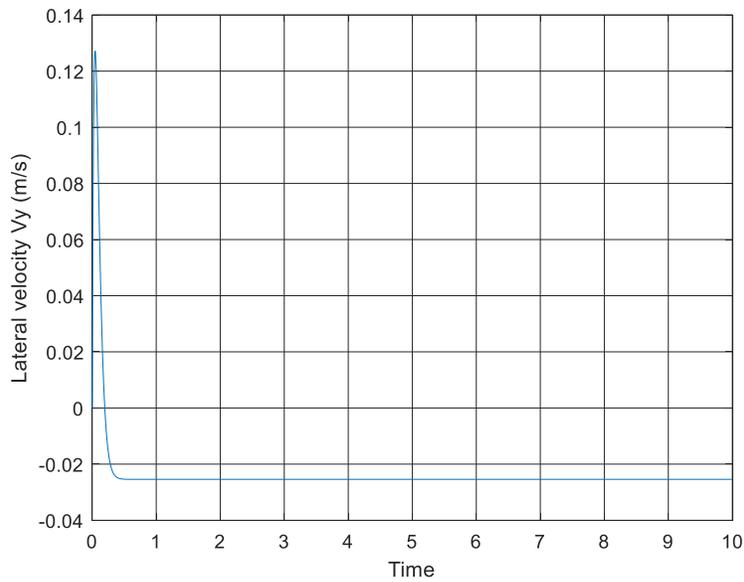


Figure 38 Lateral velocity response Vs Time

The above plot gives lateral velocity  $V_y = -0.025$  m/s and Yaw rate  $r = 2.62$  radians per second.

Thus, the setup is completely perfect and functional for further non dimensional testing and validations.

## CHAPTER 6 PI ANALYSIS

The model considered here is planar dynamics model with steering input applied on the front wheels. The bicycle model is assumed to be linear and effects of non-linearity is not under the scope of this work. In order to test dynamics similitude, Buckingham Pi theorem is used. The initial step is to collect vehicle parameters and grouping into non dimensional parameters called as  $\pi$  parameters. These grouping is done by normalizing mass, time and length scales by scaling factors dependent directly on vehicle mass, length and velocity(U).

The grouping of  $\pi$  parameters can be done after normalizing the length and time coordinates. The normalization is done by the factor like time taken by the vehicle to travel its own length at a velocity U or with time L/U.

So, the  $\pi$  parameters we get for planar dynamics could be,

$$\Pi 1 = \frac{a}{L}$$

$$\Pi 2 = \frac{b}{L}$$

$$\Pi 3 = \frac{C_f L}{mU^2}$$

$$\Pi 4 = \frac{C_r L}{mU^2}$$

$$\Pi 5 = \frac{I_{zz}}{mL^2}$$

Each vehicle with certain vehicle parameters can have certain PI values. In order to match the dynamic similitude, the full-sized vehicle should also carry the same PI values. Close matching of these non-dimensional parameters is advised. Just in case, if there is a difference in the values, there is always a greater possibility in changing the scaled vehicle parameters to match the full-sized vehicles non dimensional parameters with less effort and cost.

Sometimes there may be discrepancies in results due to some extra effects such as tire lag, suspension components, roll dynamics etc. In this project, suspension components effects are eliminated by replacing the existing suspension with a suspension rod made up of ABS and roll dynamics is not considered.

The key parameters which helps in matching the dynamics is cornering stiffness. However, the Pi values associated with this is dependent on longitudinal velocity U. Hence the velocity U must be scaled to match the full-sized vehicle.

For example,

$$\text{Let } \Pi_3 = \frac{C_f L}{mU^2} = 1 \text{ for scaled vehicle}$$

$$\text{Having } C_f = 100 \text{ N/rad, } m = 4.5 \text{ kg, } L = 0.4 \text{ m}$$

Leaves us longitudinal velocity of 2.98 m/s or 10.7 Km/h

$$\text{Matching a full-sized vehicle having same } \Pi_3 = \frac{C_f L}{mU^2} = 1$$

$$\text{Having } C_f = 80000 \text{ N/rad, } m = 1100 \text{ Kg, } L = 2.5 \text{ m}$$

Gives longitudinal velocity of 13.48 m/s or 48.5 Km/h

Hence, the dynamic effect generated in the scaled vehicle at 10.7 km/h will be equal to the full-sized vehicle at 48.5 km/h. The velocity or mass or stiffness can be changed to user desirable value to get required PI value.

State space vehicle dynamics model can be expressed as

$$\dot{X} = AX + BU$$

$$\dot{Y} = CX + DU$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_f + C_r}{mU} & \frac{C_f + C_r}{m} & \frac{bC_r - aC_f}{mU} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bC_r - aC_f}{I_{zz}U} & \frac{aC_f - bC_r}{I_{zz}} & -\frac{a^2C_f + b^2C_r}{I_{zz}U} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & \frac{C_f}{m} & 0 & \frac{aC_f}{I_{zz}} \end{bmatrix}^T$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The above state space can be completely represented in  $\pi$  parameters as below (Brennan & Alleyne, 2001a),

$$A^* = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\pi_3 - \pi_4 & -\pi_3 + \pi_4 & \pi_2\pi_4 - \pi_1\pi_3 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\pi_2\pi_4 - \pi_1\pi_3}{\pi_5} & \frac{\pi_1\pi_3 - \pi_2\pi_4}{\pi_5} & -\frac{\pi_1^2\pi_3 + \pi_2^2\pi_4}{\pi_5} \end{bmatrix}$$

$$B^* = \begin{bmatrix} 0 & \pi_3 & 0 & \frac{\pi_1\pi_3}{\pi_5} \end{bmatrix}^T$$

This modified state space model replicates the response of vehicles having two different scales. The behavior of this state space model replicates the dynamics of both the scaled and full-sized vehicles carrying the same  $\pi$  values.

# CHAPTER 7 EXAMPLES

## 7.1.1 PID CONTROL

The setup can be used to apply controls over steering and throttle. Giving throttle control to the microcontroller is a risky approach, as software glitch can make the car uncontrollable. To expand the functions of the setup, a PID control strategy is applied on the Arduino to drive the car at constant yaw rate leaving the throttle to the user. This approach expands the user's motto to do steady state drifting, when microcontroller can take significant amount of throttle control.

As in Arduino, the replication of PID control is an easier task. There are many control libraries as well, but that doesn't make much difference in writing the own PID code on the IDE.

Problem statement:

Controlling the steering angle to produce user defined constant yaw rate of the vehicle. Throttle in this case is with the user.

The vehicle should reduce the steering angle with increasing in speed, which should be verified visually during testing.

Arduino IDE code for PID control is given below;

The below Arduino code is a simple example of PID control using the setup for constant yaw rate control. The PID tuning parameter are given as kp, ki , kd whose values can be changed for better control.

```
*****
```

```
//Mpu library
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
//servo library
#include <Servo.h>
Servo myservo; // create servo object to control a servo
unsigned long lastTime=0;

MPU6050 accelgyro;
int16_t ax, ay, az; // define accel as ax,ay,az
int16_t gx, gy, gz; // define gyro as gx,gy,gz
double rax, ray, raz; // define raw acc values
double rgx, rgy, rgz; // define raw gyro values

//Define Variables we'll be connecting to, Setpoint in rad per second
double Setpoint=0, Input=0, Output=0, poutput=0, ioutput=0, doutput=0;
int steer_offset=94;

//Specify the links and initial tuning parameters
double kp=1,ki=0, kd=0;
double dt=0.0000;

void setup()
{
  Wire.begin(); // join I2C bus
  Serial.begin(115200); // initialize serial communication
  accelgyro.initialize();
  myservo.attach(9);//Attach PWM 9 Pin to servo motor
}

void loop() {
  unsigned long t=millis();
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // read measurements from device
  rgz= gz/131.0;
  Input =((rgz)*3.14)/180;

  //removing decimals from four to two
  double q=rad_rgz*100;
  int w=q;
  q=w;
  Input=q/100;
```

```

//
dt=(t-lastTime);//in milli seconds
dt=dt/1000; // in seconds

//PID CONTROLLER
/*Compute all the working error variables*/
error = Setpoint - Input;
Iout = last_Iout+((last_error+error)*dt); // change to ms from seconds
Dout = (error - last_error) / (dt);// dt in ms from seconds

/*Compute PID Output*/
poutput=kp * error;
ioutput=ki * Iout;
doutput=kd * Dout;
Output = (poutput) + (ioutput) + (doutput);

int tiredeg= Output*180/3.14;
int deg_output=tiredeg;

// limit pid values to maximum values
if(tiredeg>27)
{
    deg_output=27;
}
if(tiredeg<-27)
{
    deg_output=-27;
}

lastTime = t;
deg_output=deg_output*84/54;

pos=steer_offset+ last_deg+deg_output;
last_deg=deg_output;
myservo.write(pos);      // tell servo to go to position in variable 'pos'

}
*****

```

## 7.1.2 PID TUNING CIR/CUIT

The tuning can be made simpler using potentiometers instead of flashing the program every time to change the tuning parameters. The below code captures the user input for Kp, Ki, Kd values from the potentiometer and uses it for PID control.

```
*****
//Mpu library
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

//servo library
#include <Servo.h>
Servo myservo; // create servo object to control a servo
unsigned long lastTime=0;

MPU6050 accelgyro;
int16_t ax, ay, az; // define accel as ax,ay,az
int16_t gx, gy, gz; // define gyro as gx,gy,gz
double rax, ray, raz; // define raw acc values
double rgx, rgy, rgz; // define raw gyro values
char m[100];

//Define Variables we'll be connecting to, Setpoint in rad per second
double Setpoint=0, Input=0, Output=0, poutput=0, ioutput=0, doutput=0;

int steer_offset=94;

//Specify the links and initial tuning parameters
double kp=1,ki=0, kd=0, N=5;
int ptm0 = A0; //Kp
int ptm1= A1;//ki
int ptm2=A2;//kd
double pm0=0, pm1=0,pm2=0,pm3=0;
int pos=0;
int last_pos=0;
double last_Input=0;
double rad_rgz=0;
double last_error=0,error=0;
double last_Iout=0,Iout=0,Dout=0;
double dt=0.0000;
int last_deg=0;

void setup()
{
  Wire.begin(); // join I2C bus
  Serial.begin(115200); // initialize serial communication
  accelgyro.initialize();
```

```

myservo.attach(9);//Attach PWM 9 Pin to servo motor
}

void loop() {

//Kp mapping 0 to 0.5
kp=0;
float pm0 = analogRead(ptm0);
kp=(float)pm0*0.5/1024;

//Ki mapping 0 to 5
ki=0;
double pm1 = analogRead(ptm1);
ki=(float)pm1*5/1024;

//kd mapping 0 to 1
kd=0;
double pm2 = analogRead(ptm2);
kd=(float)pm2*1/1024;
unsigned long t=millis();

accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // read measurements from device
rgz= gz/131.0;
Input=((rgz)*3.14)/180;

//removing decimals
double q=rad_rgz*100;
int w=q;
q=w;
Input=q/100;
dt=(t-lastTime)/1000;//in seconds

//PID CONTROLLER
/*Compute all the working error variables*/

error = Setpoint - Input;
Iout = last_Iout+((last_error+error) *dt);
Dout = (error - last_error) / (dt);

/*Compute PID Output*/
poutput=kp * error;
ioutput=ki * Iout;
doutput=kd * Dout;
Output = (poutput) + (ioutput) + (doutput);

int tiredeg= Output*180/3.14;
int deg_output=tiredeg;
//Limiting the maximum value from PID
if(tiredeg>27)

```

```

{
  deg_output=27;
}
if(tiredeg<-27)
  { deg_output=-27;
  }
lastTime = t;
deg_output=deg_output*84/54;
pos=steer_offset+ last_deg+deg_output;
last_deg=deg_output;
myservo.write(pos);          // tell servo to go to position in variable 'pos'
  last_error = error;
  last_Iout=Iout;
  last_DFout = DFout;
  last_pos=pos;
}
*****

```

The wiring diagram for this extended setup is given below:

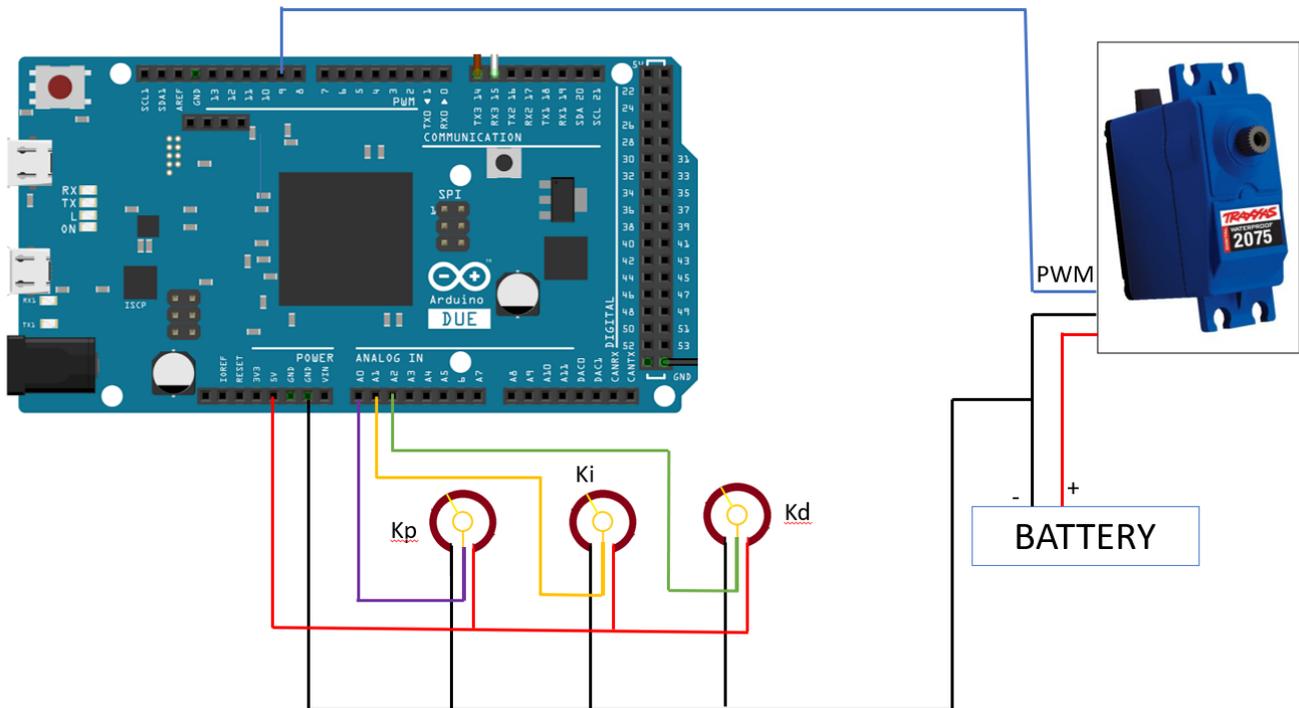
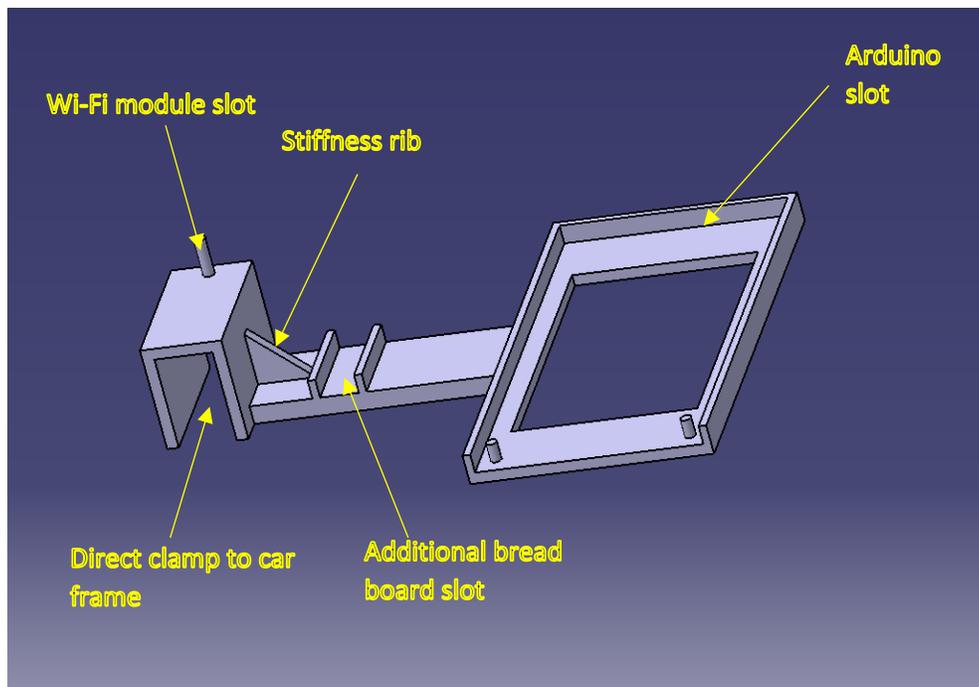


Figure 40 Arduino Due, Steering Servo , Potentiometers wiring diagram - PID tuning

## CHAPTER 8 SUPPORT STRUCTURE

A support structure is 3D printed in RMIT Bundoora facility to accommodate all the electronics components. The image of the structure is given below.

Material used: ABS Weight:38g



*Figure 41 Support structure CAD model*

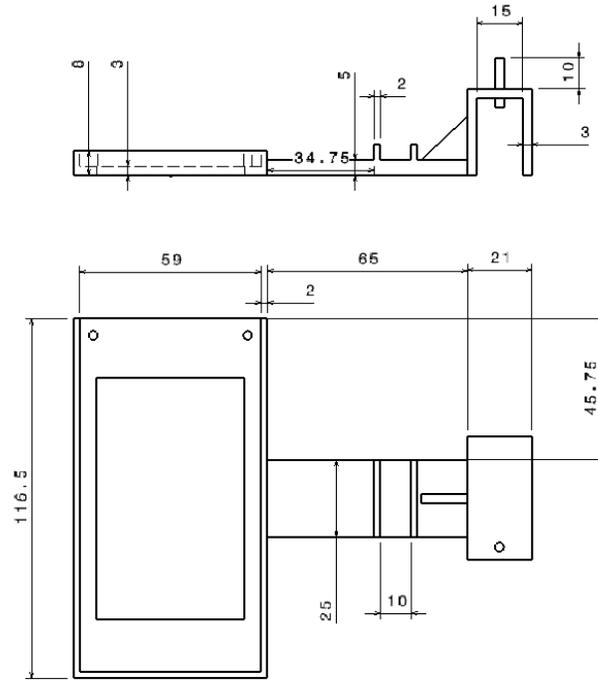


Figure 42 Technical 2D Diagram - Support structure (All dimensions in mm)

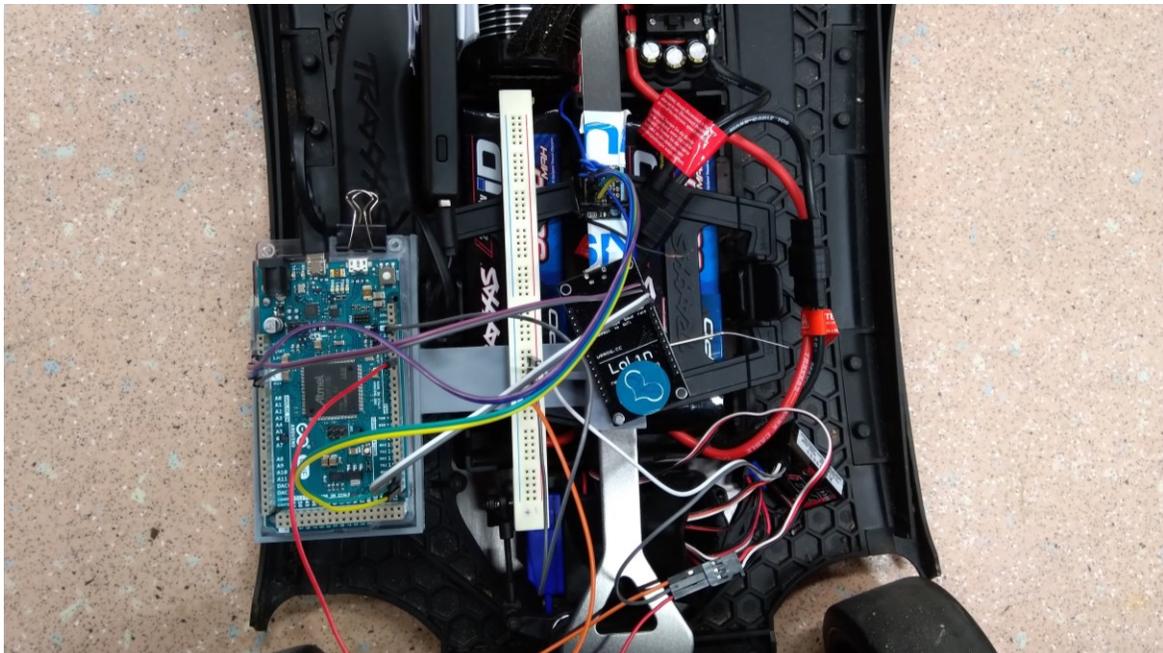


Figure 43 Complete setup with Arduino Due, Wi-Fi modules, Sensor , Power Supply, Steering servo with Support structure assembled on the R/C car

## CHAPTER 9 CONCLUSION

Vehicle testing from the Original Equipment Manufacturers side is really a tough task. Because it includes, lot of time, hardware, software, man power etc. Testing includes indoor as well as outdoor testing. Comparably outdoor testing consumes more money and time than indoor ones. This project helps those to do the outdoor testing indoors. With the help of Pi theorem and few electronic modules it is possible and it is proven by this work.

The project objective in building a setup to do planar dynamics validations indoors is fairly achieved in this work. The setup is equipped with sensors and digital data acquisition system and control strategies. This work is an example that indoor testing is even possible in low cost. The setup is capable of logging data wirelessly. When a manufacturer decides to test the vehicle, the one major thing required is to develop a scaled model of the vehicle to be tested or grab an already available scaled model and do changes to resemble the vehicle which is to be tested.

The setup is built by formulating the requirements to do planar dynamics validations. The requirements were properly listed and things required are selected from the available market. The important parameters required to be collected from the vehicle is acceleration on three axes and angular velocity on three axes. For planar dynamics study these parameters are enough. The number of parameters to be read is not at all restricted. Many sensors can be implemented in this setup as it is designed in a way to expand the features easily. For sensors, many types of sensors available in the market is shortlisted and based on the features and connection types the best matching one is filtered. The requirement from the sensor side is that it should have higher resolution, low power consumption, less latency, advanced communication types and simpler to handle. The sensor MPU 6050 chosen has fulfilled all the above requirements. It has 16bit resolution of

data with inbuilt analog to digital convertor. It really works at low power from 3V to 5 V and consumes very less steady leakage current. The communication type is I2C which is one of the advanced communication types with less latency and capable of handling more devices (around 1008) at the same time, which is a preferable feature just in case in future if the setup requires more modules to get embedded. The wiring and reading the sensor is easier and simpler.

The microcontroller chosen is Arduino Due here. The main preferred feature of microcontroller for this setup is that it should be capable of handling more devices, faster clock range, easy compatibility and less software glitches. Arduino due fulfilled the requirements having fastest clock frequency of 84 MHz and 54 pins expand the setup and a steady standard firmware by Arduino. The transmission part of the data acquisition system works at 100Hz which can be pushed up to 200Hz. So far, the modules used in the setup are two ESP 12 Wi-Fi modules. The preferred features in setting up the Wi-Fi is the network should agree with Local Interconnect Network architecture to avoid collisions and data loss. The transfer protocol used should be capable of handling contention and data rate should be acceptable. The ESP 12 used in this setup fulfills all the requirements by obeying LIN architecture, TCP stack protocol and working data rate of 100Hz. The additional fruity feature of this module is that, ESP 12 itself is a mini microcontroller and can do miniature control or reading tasks at max 160 MHz (higher than Arduino which makes data contention impossible) provide the task is not overfilling the module's RAM. It could be a helping hand while expanding the setup. Both the transmitter and receiver uses ESP 12 modules which facilitates the compatibility and reduces complexity in programming it. The Wi-Fi network created in a software enable Wi-Fi network which extends the possibility to directly connect other devices such as PC or Tablet as well to receive data. The data received from the receiver side is decoded with the help of opensource script communicator which is 100% user friendly and has a console to modify and program the received data as per user's wish. The interface uses java language which is notable.

Step by step testing of devices, system and the setup is done over the period of time. Each and every device is tested and properly optimized for its maximum safe conditions. In order to validate the setup, a scaled car, Traxxas X01 was chosen and the setup is loaded and initially tested on the ground. Since planar dynamics is our interest, so the suspensions of the car were replaced with rods. Structure which accommodates all the modules was 3D printed and properly places in the car. The complete electronic setup to be placed in the car weighs only up to 100 g which is significantly less when compared to the weight of the car.

The car is tested on the ground for proper working of the data acquisition system and suspension rods. Later the car was subjected to many maneuvers like low speed steady cornering and high-speed steady cornering. The data were logged in the PC, filtered, smoothed using MATLAB and the behavior of the vehicle was noted. As per the results the vehicle after removing the suspensions completely behaved as a rigid vehicle body and the results were shown. In order to verify it, it is mandatory to cross check it with known theoretical models. In this case it was bicycle model. But the verification required some physical parameters such as mass, COG location, Mass moment of inertia and cornering stiffness. Each parameter was properly measured with laboratory equipment and the values were reported. Mass moment of inertia was measured using Bifilar pendulum technique and the steps and calculations were shown. Cornering stiffness of the vehicle was derived with the use of sensor data at high speed steady state cornering maneuver. The steps followed and the results were reported and verified. With the help of measured parameters, the setup was verified against steady state of bicycle model and the results were reported.

Experiments showed that the setup is completely perfect and functional for further non dimensional testing and validations and it was found that the car behaves linear.

After validating the setup, the steps to perform non dimensional validations were reported. The equations, pi parameters and mapping with full sized car was explained clearly. The extension or expansion of the setup like controlling the steering servo motor, throttle and applying Proportional-Integral-Derivative control over steering to control yaw rate etc., were explained with example program sketches and wire diagrams. Another separate example for easier tuning of PID control with potentiometers was explained with sketches and wire diagram.

To save the components from throwing out of the car during high speed testing, a support structure was developed using CATIA and 3D printed. The material used was ABS. Th components are then properly clamped on the structure to avoid disconnections due to vibrations and G forces during testing.

In concluding the project, the setup developed is versatile and has the following features:

1. 3 axes Acceleration and Angular velocity logging at 100Hz and easy to push up to 200Hz
2. 16bit data resolution and expandable circuit with more sensors, controllers, displays etc., up to 1008 devices
3. Programmable console for data logging at user level
4. Control of steering motor and throttle based on the user requirement
5. PID control application
6. Inbuilt noise filtering, low latency, low cost and low powered
7. Variable COG and mass
8. Possible testing at user desirable high speeds
9. Easily diagnosable and Ready to use

# REFERENCES

- (Bekker; Brennan & Alleyne, 2001a, 2001b; Buckingham, 1914; Buyanov, 1988; Emori, 1969; Fujimoto & Takahashi, 2006; Genta & Delprete, 1994; Jazar, 2017, 2019; Kalinowski, 2013; Lapapong et al., 2009; Lian et al., 2015; Nuttal, 1951; O'Brien et al., 2004; Polley, Alleyne, & Vries, 2006; Polley & Alleyne, 2004; Segel, 1956; Sierra et al., 2006)
- Arduino. (2019). Arduino. [www.Arduino.cc](http://www.Arduino.cc).  
GitHub [www.github.com](http://www.github.com)
- Bekker, M. Introduction to terrain-vehicle systems. 1969. *Ann Arbor, MI: University of Michigan Press*.
- Brennan, S., & Alleyne, A. (2001a). Robust scalable vehicle control via non-dimensional vehicle dynamics. *Vehicle System Dynamics*, 36(4-5), 255-277.
- Brennan, S., & Alleyne, A. (2001b). Using a scale testbed: Controller design and evaluation. *IEEE Control Systems Magazine*, 21(3), 15-26.
- Buckingham, E. (1914). On physically similar systems; illustrations of the use of dimensional equations. *Physical review*, 4(4), 345.
- Buyanov, E. (1988). Method and apparatus for accurate determination of the inertia tensor of a solid body. *Measurement techniques*, 31(12), 1181-1184.
- Emori, R. (1969). Automobile accident reconstruction. *UCLA Motor Vehicle Safety Contract Final Report*, 69-51.
- Fujimoto, H., & Takahashi, N. (2006). Cornering stiffness and slip angle estimation of electric vehicle for direct yaw-moment control. *IFAC Proceedings Volumes*, 39(16), 496-501.
- Genta, G., & Delprete, C. (1994). Some considerations on the experimental determination of moments of inertia. *Meccanica*, 29(2), 125-141.
- Jazar, R. N. (2017). *Vehicle dynamics: theory and application*: Springer.
- Jazar, R. N. (2019). *ADVANCED VEHICLE DYNAMICS*: Springer.
- Kalinowski, J. (2013). Conversion of a Formula SAE Vehicle to Full Drive-by-Wire Capability. *REV Project, UWA*.
- Lapapong, S., Gupta, V., Callejas, E., & Brennan, S. (2009). Fidelity of using scaled vehicles for chassis dynamic studies. *Vehicle System Dynamics*, 47(11), 1401-1437.
- Lian, Y., Zhao, Y., Hu, L., & Tian, Y. (2015). Cornering stiffness and sideslip angle estimation based on simplified lateral dynamic models for four-in-wheel-motor-driven electric vehicles with lateral tire force information. *International Journal of Automotive Technology*, 16(4), 669-683.
- MCU, N. (2019). Node MCU- DEV KIT. [www.nodemcu.com](http://www.nodemcu.com).
- Nuttal, C. (1951). Preliminary Study of the Turning Characteristics of Articulated Tracked Vehicles. *Hoboken, NJ: Experimental Towing Tank, Stevens Inst. of Technology*.
- O'Brien, R. T., Piepmeier, J. A., Hoblet, P. C., Burns, S. R., & George, C. E. (2004). *Scale-model vehicle analysis using an off-the-shelf scale-model testing apparatus*. Paper presented at the Proceedings of the 2004 American Control Conference.
- Polley, M., Alleyne, A., & Vries, E. D. (2006). Scaled vehicle tire characteristics: dimensionless analysis. *Vehicle System Dynamics*, 44(2), 87-105.
- Polley, M., & Alleyne, A. G. (2004). *Dimensionless analysis of tire characteristics for vehicle dynamics studies*. Paper presented at the Proceedings of the 2004 American control conference.
- Schiller, C. (2019). Vehicle inertia measuring machine. <http://www.cfm-schiller.de>.

Segel, L. (1956). Theoretical prediction and experimental substantiation of the response of the automobile to steering control. *Proceedings of the Institution of Mechanical Engineers: Automobile Division*, 10(1), 310-330.

Sierra, C., Tseng, E., Jain, A., & Peng, H. (2006). Cornering stiffness estimation based on vehicle lateral dynamics. *Vehicle System Dynamics*, 44(sup1), 24-38.

Traxxas. (2019). Traxxas. [www.traxxas.com](http://www.traxxas.com).

Spark fun [www.learn.sparkfun.com](http://www.learn.sparkfun.com)

Logic Voltage [www.logicvoltage.com](http://www.logicvoltage.com)

[www.arduino.stackexchange.com](http://www.arduino.stackexchange.com)