MISR UNIVERSITY OF SCIENCE AND TECHNOLOGY

College of Engineering and Technology

Department of Mechatronics Engineering

B. Eng. Final Year Project

Autonomous vehicle

By Group (2):

Abdelrhman Khairy Ahmed	58312
Ahmed Mohamed Ibrahem	58559
Hazem Khaled Mahmoud	58515
Norhan Tawfik Rady	64767
Omar Ashraf Mohamed	58612
Shady Fayez Nassif	58644

Supervised By:

Prof. Farid A.Tolba ----------

Prof. Mohamed Hamdy ----------

Prof. Mohamed Ibrahim ----------

Prof. Mostafa Eissa ----------

Prof. Bahaa Nasser ----------

Prof. Ramadan El Souedy ----------

Table of content

Table of content 2

List of figures 9

Dedication 12

Acknowledgement 13

Abstract 14

Chapter 1 15

Introduction and motivation 15

1- What are autonomous vehicles. 16

2- Brief History and Current State of Autonomous Vehicles: 16

3- Layers of autonomy: 17

4- Technology behind autonomous vehicles: 18

5- Autonomous Vehicle Market: 19

6- Project objective: 20

Chapter 2 23

Reverse engineering and improvement 23

1- What Is Reverse Engineering: 24

2- How does reverse engineering work: 24

3- Initial specification of ATV: 25

4- Improvement specification: 26

4.1- Software control of speed and acceleration: 26

4.2- Hardware control of speed:- 27

4.3- Software control of brake: 27

4.4- Hardware control of brake: 27

4.5- Software control of steer angle: 28

5- Why use linear actuator: 28

5.1- Hardware control of steer angle: 29

5.2- Installed sensor: 29

Chapter 3 30

Mechanical Improvement 30

1- An Overview of X-By Wire Systems: 31
2- Brake-By Wire system: 31
2.1- The mechanism on SOLIDWORK: 32
2.2- Components: 32
2.3- Manufacturing processes: 33
2.4- Why the material of the roller coupler is steel? 33
2.5- Motor specification: 33
2.6- Roller specification: 33
2.7- Roller coupler specification: 33
3- Steer-By wire system: 34
3.1- Components 34
3.2- Manufacturing processes: 35
3.3- Linear Actuator specification: 35
3.4- End ball bearing specification: 35
4- ATV Dynamic Analysis 37
4.1- Road Loads 37
5- Determination of Steering Torque 39
Chapter 4 41
Modeling and Control 41
1- DC-Motor modeling 42
1.1- DC-Motor Parameters: 42
1.2- DC-Motor Equations: 43
1.3- Parameters acquiring: 44
2- Feedback and Control 45
2.1- Why use a feedback control: 45
2.2- PID Controller: 47
2.3- Choosing a controller: 49
2.4- Tuning the controller: 49
2.5- PI-Controller Implementation: 50
2.6- Conclusion: 50

Chapter 5 51
Embedded System 51
1-Embedded System 52
Embedded System hardware: 52
2-History of Embedded System: 52
3-What is Microcontrollers? 53
AVR Microcontroller : 53
4-Why ATmega32 Not Arduino 53
5- Microcontroller abstraction layer: 56
5.1- DIO (Digital Input Output) 56
5.2- Interrupt 57
5.3- Timer and counter 57
5.4- Pulse Width Modulation: 59
5.5- UART (Universal Asynchronous Receiver Transmitter) 61
6-DC Motor Speed 62
6.1- Read motor speed 62
6.2- Control the motor speed 63
6.3- Connect the motor to the microcontroller using BTS7960 driver 63
7- Steering Angle 64
Chapter 6 66
Motion planning control 66
1- Introduction: 67
2- ATV Steering system: 67
2.1- Ackermann steering: 67
3- Kinematics model: 69
3.1- Forward kinematics model of Ackermann: 69
4- Path model: 71
5- Pose Errors: 72
6- Controller design: 73
7- Model predictive controller: 74

8- Control block diagram: 76
9- Results: 77
Chapter 7 79
Localization 79
Introduction for localization 80
1- MPU 9250 80
1.1- MPU 9250 Specifications 81
1.2- MPU and Arduino hardware connection 82
1.3- Code explanation 83
2- NEO-6M GPS 85
2.1- GPS specifications 85
2.2- GPS Features 86
2.3- Hard ware connection 86
2.4- Code explanation 87
2.5- UTM (Universal Transverse Mercator) 89
3- Odometry 91
4- Localization by sensor fusion 92
4.1- Why we use imu with gps by sensor fusion not gps only? 92
4.2- GPS and IMU complement each other by: 93
4.3- How to achieve sensor fusion between imu and GPS? 93
5- Kalman filter 94
5.1- What is a kalman filter? 94
5.2- Why did we use a Kalman filter? 94
5.3- How Kalman filter work? 94
5.4- Extended kalman filter 95
5.5- Parameters for EKF 96
5.6- Filter design 96
5.7- How calculated parameters 97
5.8- Motion Model 97
5.9- Observation Model 97

The robot can get x-y position infomation from GPS 97
5.10- Extented Kalman Filter equations in code 98
Chapter 8 99
Path Planning 99
1- Introduction 100
2- Map 101
3- Searching Algorithms 103
3.1- Dijkstra's algorithm: 103
3.2- A* algorithm: 103
3.3- Why A*? 104
4- What is heuristic function? 104
4.1- Manhattan distance: 105
4.2- Diagonal distance: 105
4.3- Euclidean distance: 106
4.4- What is the problems of A* searching algorithm? 106
4.5- A*hybrid: 106
4.6- Conclusion: 107
Chapter 9 109
Computer Vision 109
1- Introduction 110
2- Lane finding 111
Appendix 115
Components of robots: 115
Final design: 115
Final design without box: 118
Base of robot: 123
Box for compounds: 124
Steering shaft: 124
Base motor for motion: 126
Linear actuator: 126

Chain disc: 127
Chain: 127
Encoder_1: 128
Encoder_2: 128
Shaft for back wheel: 128
Flexible coupler: 129
Break disc: 129
Base for steering shaft: 130
Encoder base: 130
Wheel: 131
Rod: 132
Ball joint: 133
Final design drawing: 134
1) Base of robot: 135
2) Shaft for back wheel : 136
3) Wheels: 137
4) Shaft steering 138
5) Stand shaft steering 139
6) Base steering 140
7) Steering rod 141
8) Rod connecting 142
9) Balljoint 143
10) Break circle 144
11) chain circle 145
12) chain_1 146
13) Chain_2 147
14) Flexible coupler 148
15) Base encoder_1 149
16) Base encoder_2 150
17) Box for compounds: 151

18) Roller..... 152
19) Roller Coupler..... 153
References..... 154

List of figures

Figure1. 1: History of self driving car 16
Figure1. 2: Autonomous levels 17
Figure1. 3: Perception sensor 18
Figure1. 4: self driving car copmanys 19
Figure1. 5: project steps 20
Figure1. 6: lane finding 21
Figure1. 7: Object Detection and recognition 21
Figure1. 8: ADAPTIVE CRUISE CONTROL 22

Figure2. 1: reverse engineering steps 24
Figure2. 2 solid works atv 25
Figure2. 3: Acceleration by wire 26
Figure2. 4: steer angle software 28

Figure3. 1: (Hand brake) 31
Figure3. 2:(Brake disc) 31
Figure3. 3:atv mechanism 32
Figure3. 4:(the steering mechanism) 34
Figure3. 5:(Linear actuator) 34
Figure3. 6: (End ball bearing) 34
Figure3. 7:(STEERING mechanism) 36
Figure3. 8: (steering mechanism) 36
Figure3. 9 (Road loads) 37
Figure3. 10: (CG from rear and front) 39
Figure3. 11: (Forces on Tire) 39

Figure4. 1: Dc-Motor 42
Figure4. 2: DC Motor transfer function 44
Figure4. 3: Output Response 45
Figure4. 4: Open Loop system 45
Figure4. 5: Close Loop system 46
Figure4. 6: Steady state error 46
Figure4. 7: PI-controller Response 47
Figure4. 8: PI-Controller 48
Figure4. 9: PID-Controller Response 48
Figure4. 10: PID-Controller 49
Figure4. 11: PI-controller using MATLAB SIMULINK 49
Figure4. 12: System response 50

Figure5. 1: ATmega32 53
Figure5. 2: Pinout ATmega32 55
Figure5. 3: atmega32 pins 56
Figure5. 4: interrupt. 57
Figure5. 5: timer types 58
Figure5. 6: pulse width modulation. 59
Figure5. 7: PWM period. 60
Figure5. 8: PWM duty cycles 60
Figure5. 9: PWM parameters. 60
Figure5. 10: UART 61
Figure5. 11: Quadrature 64
Figure5. 12: X1 Encoding 65

Figure6. 1: ATV Ackermann steering system 67
Figure6. 2: Side-pivot steering with parallel-set track-rod arms 68
Figure6. 3: SIDE-PIVOT STEERING WITH INCLINED TRACK-ROD ARMS 68
Figure6. 4: kinematic bicycle model 69
Figure6. 5: math vehicle model 70
Figure6. 6: Path modeling 71
Figure6. 7: the cross-track error (CTE) 72
Figure6. 8: the heading error 72
Figure6. 9: mpc operation 74
Figure6. 11: MPC Prediction Horizon 76
Figure6. 12: MPC Control block diagram 76
Figure6. 13: Position Real-Time Plot 77

Figure7. 1: MPU Orientation of Axes of Sensitivity and Polarity of Rotation for Accelerometer and Gyroscope. 80
Figure7. 2: ORIENTATION OF AXES OF SENSITIVITY FOR COMPASS 81
Figure7. 3: Arduino and MPU connection 82
Figure7. 4: Arduino Uno Pinout 83
Figure7. 5: MPU drift solution filter 83
Figure7. 6: MPU yaw angle at zero position before filter 84
Figure7. 7: MPU yaw angle at zero position after filter 84
Figure7. 8: NEO-6M GPS 85
Figure7. 9: GPS and TTL connection 86
Figure7. 10: U-blox GPS OUTPUT 87
Figure7. 11: U-blox GPS output 88
Figure7. 12: UTM ZONES 89

Figure7. 13: UTM Zone Numbers 89
Figure7. 14: Egypt zone axis in UTM system 90
Figure7. 15: Odometry 91
Figure7. 16: gps and imu comparsion 92
Figure7. 17: Kalman Filter process 93
Figure7. 18:kalman filter estimation 94
Figure7. 19: Kalman filter algorithm 94
Figure7. 22: Localization Process using kalman filter 98

Figure8. 1: (A) Environment with obstacles and starting point and goal configurations, and (B) one out of many possible paths from the starting point to the goal configuration 100
Figure8. 2: OSM map of part of city 101
Figure8. 3: OSM for the same city but for streets only 102
Figure8. 4: OSM that will be able to know the states of the environment at any point 102
Figure8. 5: Manhattan distance 105
Figure8. 6: diagonal distance 105
Figure8. 7: EUCLIDEAN DISTANCE 106
Figure8. 8: map in with to search 107
Figure8. 9: path that the car will follow 108
Figure8. 10: another example to the path planning 108

figure9. 1: Lane finding and road segmentation 110
figure9. 2: lane finding 111
figure9. 3 : Image Processing Comparsion 111
figure9. 4: Image Processing original and wraped 112
figure9. 5: Histogram 113
figure9. 6: Books of window search 113
figure9. 7: Window Search 114
figure9. 8: Lane keeping 114

Dedication

We are deeply indebted to our great *Prof. Farid A. Jolba* who offered his valuable time whenever we needed. His full encouragement and even more cooperation were always with us.

We are all sad for his loss and hope our project that we dedicated for him can make him proud of us.

Acknowledgement

Alhamdulillah, first of all we would like to thanks God.

This work would not have been possible without the support of our supervisors, advisors, and teaching assistants.

We are extremely grateful for the opportunity to gain lots of experience in real time projects, followed by the knowledge of how to design and analyze real projects. For that we want to thank all the people who made it possible for students like us.

Special thanks to the graduation Project Unit for the efforts they did to provide us with all useful information and making the path clear for the students to implement all the education periods in real-time project design and analysis.

Prof. Farid A. Tolbah
Prof. Mohmed Hamdy
Prof. Mohmed Ibrahim
Prof. Mostafa Fissa
Prof. Bahaa Nasser
Prof. Ramadan el souedy

we would like to express our sincere gratitude for their Valuable time and efforts during the project. Special thanks to their support and co-operation in our difficult times.

Great thanks to our project guide *Eng. Waleed A. Al-Badry* who helped, and advised us to be in right way to achieve our goals.

Abstract

In this project we will be working on Transforming an ATV (All Train Vehicle) to a self-driving vehicle that can work in real environment.

First, we start by reverse engineering the ATV through constructing a cad design to be able to modify on it. After that we well upgrading it from a manual system to autonomies system by turning the steering wheel from manually operated to mechanically controlled through motor. Applying a motor to control the deceleration of the wheel. Applying a feedback system to though motors.

Second, modeling and controlling the motor that are responsible for both the speed of which the ATV move and being able to control the angle in which the steering wheels are positioned.

Third, using a microcontroller to being able to read the sensor data and implementing the control theory to the motor.

Finally, building a motion control system using the kinematics equation to be able to control the heading and destination in with the ATV moves to.

Chapter 1

Introduction and motivation

1- What are autonomous vehicles.

Autonomous vehicles are cars or trucks in which human drivers are never required to take control to safely operate the vehicle. Also known as self-driving or "driverless" cars, they combine sensors and software to control, navigate, and drive the vehicle.

2- Brief History and Current State of Autonomous Vehicles:

While futurists have envisioned vehicles that drive themselves for decades, research into AV technology can be divided into three phases. From approximately 1980 to 2003, university research centers worked on two visions of vehicle automation. The first were automated highways systems where relatively "dumb" vehicles relied on highway infrastructure to guide them. Other groups worked on AVs that did not require special roads. From 2003 to 2007, the U.S. Defense Advanced Research Projects Agency (DARPA) held three "Grand Challenges" that markedly accelerated advancements in AV technology. The first two were held in rural environments, while the third took place in an urban environment. Each of these spurred university teams to develop the technology.

More recently, private companies have advanced AVs. Google's Driverless Car initiative has developed and tested a fleet of cars and initiated campaigns to demonstrate the applications of the technology for example, through videos highlighting mobility offered to the blind (Google, 2012). In 2013, Audi and Toyota both unveiled their AV visions and research programs at the International Consumer Electronics Show, an annual event held every January in Las Vegas (Hsu, 2013). Nissan has also recently announced plans to sell an AV by 2020.

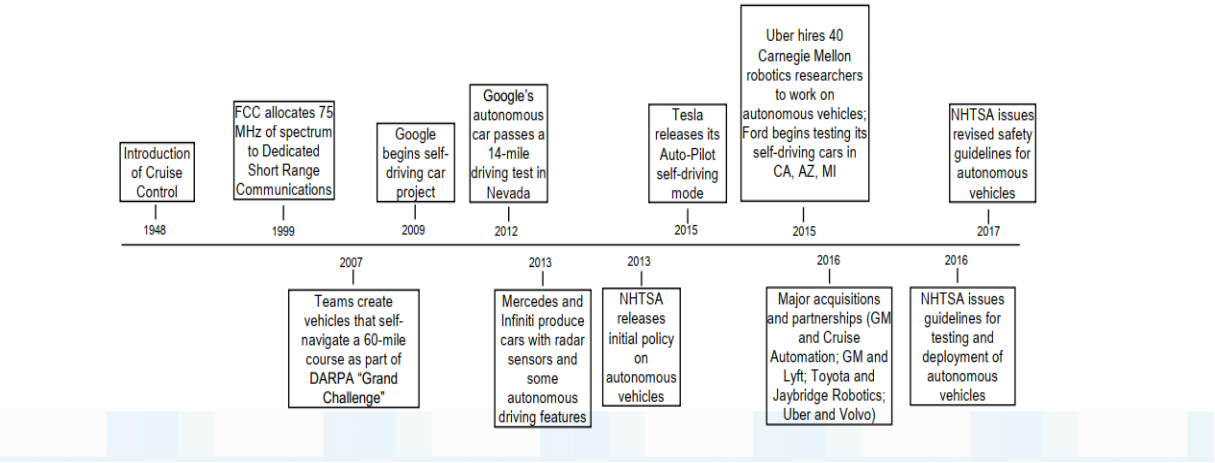


FIGURE1. 1: HISTORY OF SELF DRIVING CAR

3- Layers of autonomy:

Different cars are capable of different levels of self-driving and are often described by researchers on a scale of 0-5.

Level 0: All major systems are controlled by humans
Level 1: Certain systems, such as cruise control or automatic braking, may be controlled by the car, one at a time.
Level 2: The car offers at least two simultaneous automated functions, like acceleration and steering, but requires humans for safe operation.
Level 3: The car can manage all safety-critical functions under certain conditions, but the driver is expected to take over when alerted.
Level 4: The car is fully autonomous in some driving scenarios, though not all.
Level 5: The car is completely capable of self-driving in every situation.

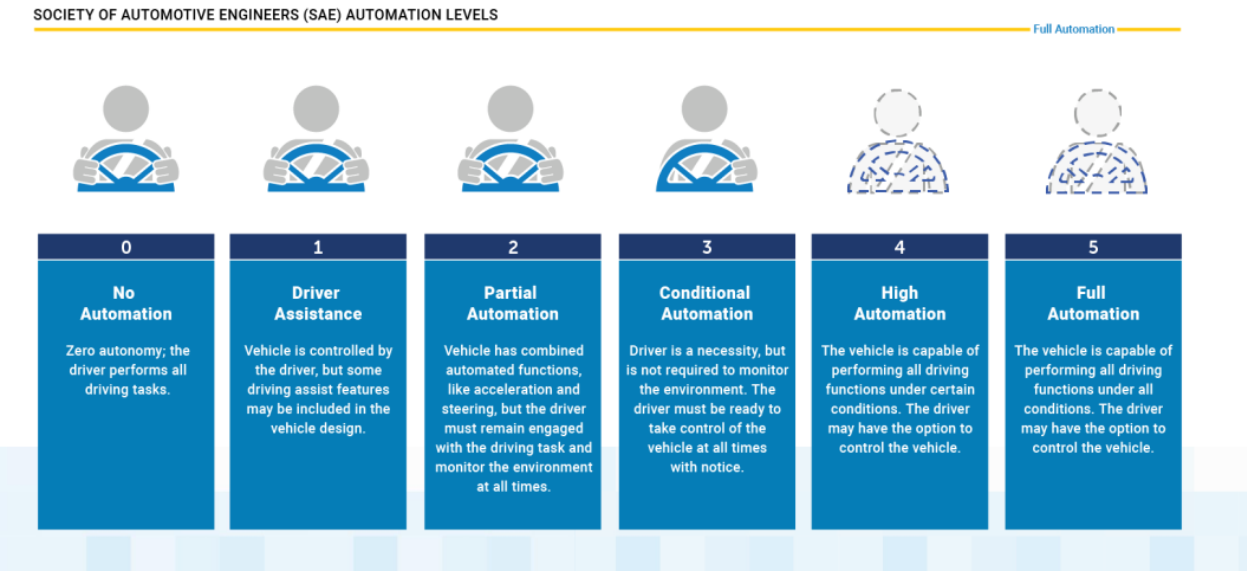


FIGURE1. 2: AUTONOMOUS LEVELS

[1]

4- Technology behind autonomous vehicles:

Self-driving vehicles employ a wide range of technologies like radar, cameras, ultrasound, and radio antennas to navigate safely on our roads.
In modern autonomous vehicles, these technologies are used in conjunction with one another, as each one provides a layer of autonomy that helps make the entire system more reliable and robust.
For example, Tesla's driverless car technology, known as "Autopilot", uses eight cameras to provide 360-degree visibility, while twelve ultrasonic sensors and a front-facing radar work to analyze the vehicle's surroundings for potential hazards.

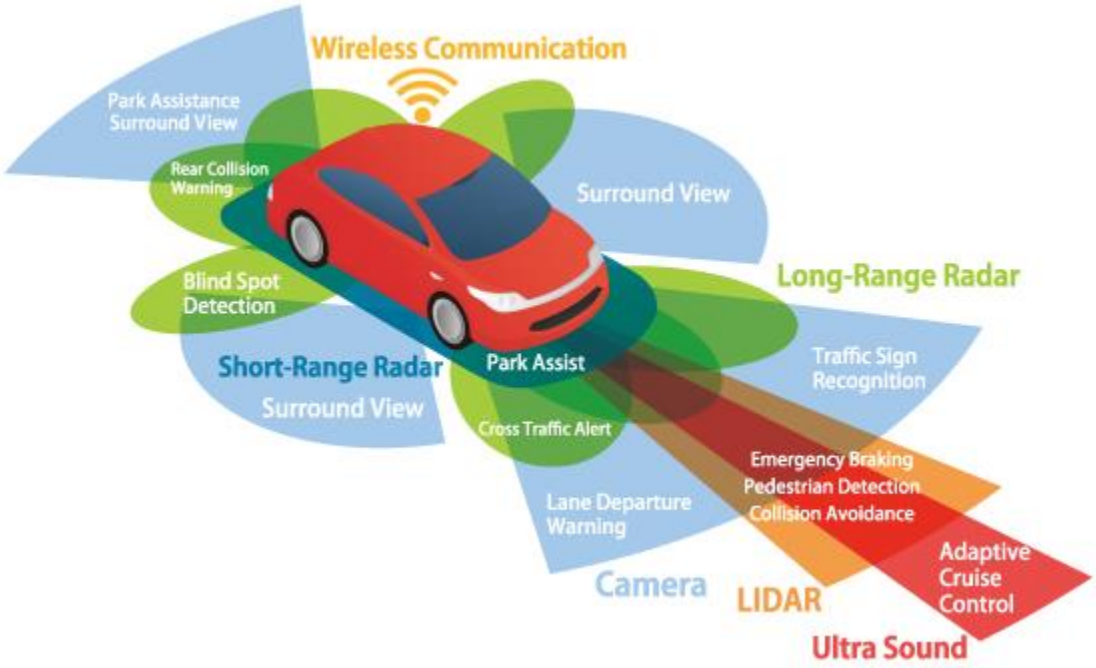


FIGURE1. 3: PERCIPTION SENSOR

[2]

Key Physical Components of Autonomous Vehicles:

Cameras – Provide real-time obstacle detection to facilitate lane departure and track roadway information (like road signs).
Radar – Radio waves detect short & long-range depth.
LIDAR – Measures distance by illuminating target with pulsed laser light and measuring reflected pulses with sensors to create 3-D map of area.
GPS – Triangulates position of car using satellites. Current GPS technology is limited to a certain distance. Advanced GPS is in development.

Ultrasonic Sensors – Uses high-frequency sound waves and bounce-back to calculate distance. Best in close range.
Central Computer – "Brain" of the vehicle. Receives information from various components and helps direct vehicle overall.
DRSC - Based Receiver – Communications device permitting vehicle to communicate with other vehicles (V2V) using DSRC, a wireless communication standard that enables reliable data transmission in active safety applications. NHTSA has promoted the use of DSRC.

5- Autonomous Vehicle Market:

The global autonomous vehicle market size is projected to be valued at $54.23 billion in 2019 and is projected to garner $556.67 billion by 2026.
The major companies profiled in the report are General Motors, Daimler AG, Ford Motor Company., Volkswagen Group, BMW AG, Renault-Nissan-Mitsubishi alliance, Volvo-Autoliv-Ericsson-Zenuity alliance, Groupe SA, AB Volvo, Toyota Motor Corporation, and Tesla Inc. Other companies in accordance with auto suppliers are Robert Bosch GMBH, Aptiv, Continental AG, and Denso Corporation. also, by technology providers, it includes Waymo, NVDIA Corporation, Intel Corporation, Baidu, and Samsung. And based on autonomous vehicle as a service provider, it includes Uber, Lyft and Didi Chuxing.

FIGURE1. 4: SELF DRIVING CAR COPMANYS

[1]

6- Project objective:

- Our goal in this project to transform an electrical ATV (All-Terrain Vehicle) to be fully autonomous, we select a small electric ATV to reduce the cost of the project but without losing the ability of navigate and testing the vehicle in real world street and highway.
- Our strategy is to split the project in two parts:
- First part is to replace the manual mechanical steering, throttle and breaking to be driven by computer software with feedback control and install sensors needed in autonomous drive mode like camera, LIDAR, encoder .etc.
- Second part is to develop the high level algorithms needed for computer vision and localization...etc.

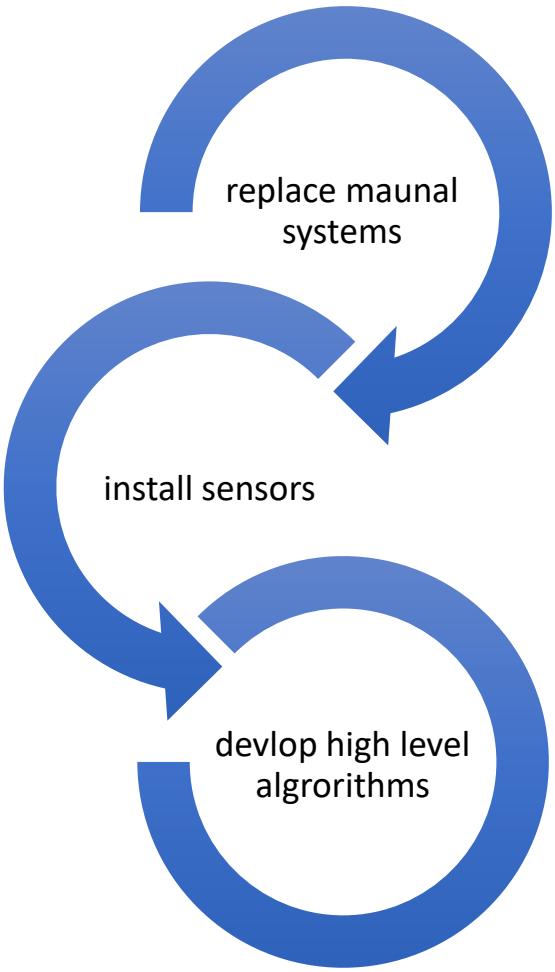


FIGURE1. 5: PROJECT STEPS

- After replacing all the manual system in the vehicle with X-by wire systems and install all needed sensors, the vehicle will have three main features:
- Lane keeping.
- Adaptive cruise control.
- Object Detection and recognition.

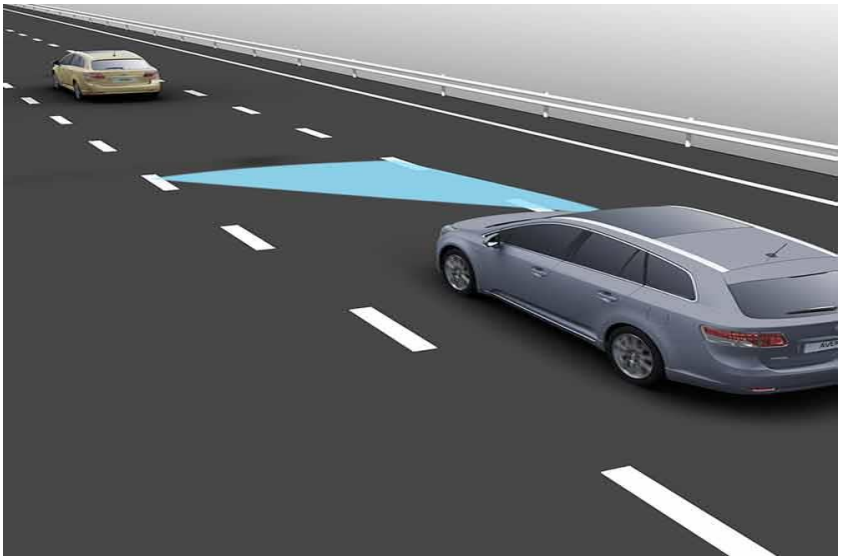


FIGURE1. 6: LANE FINDING

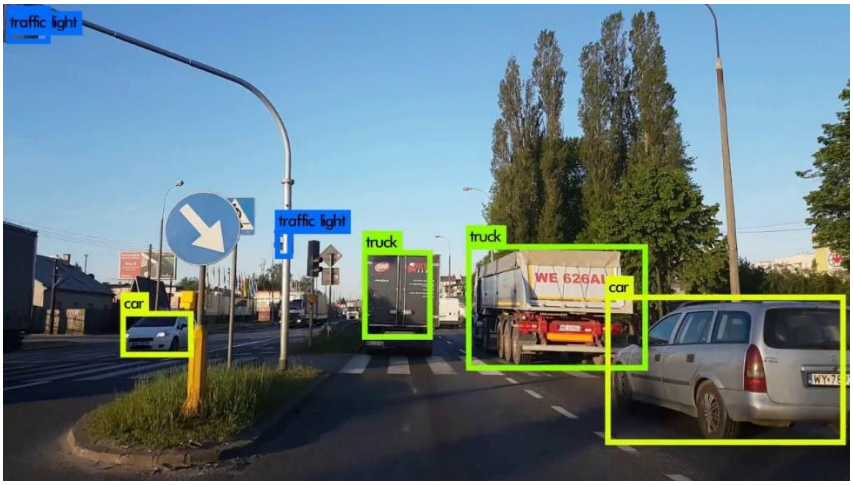


FIGURE1. 7: OBJECT DETECTION AND RECOGNITION

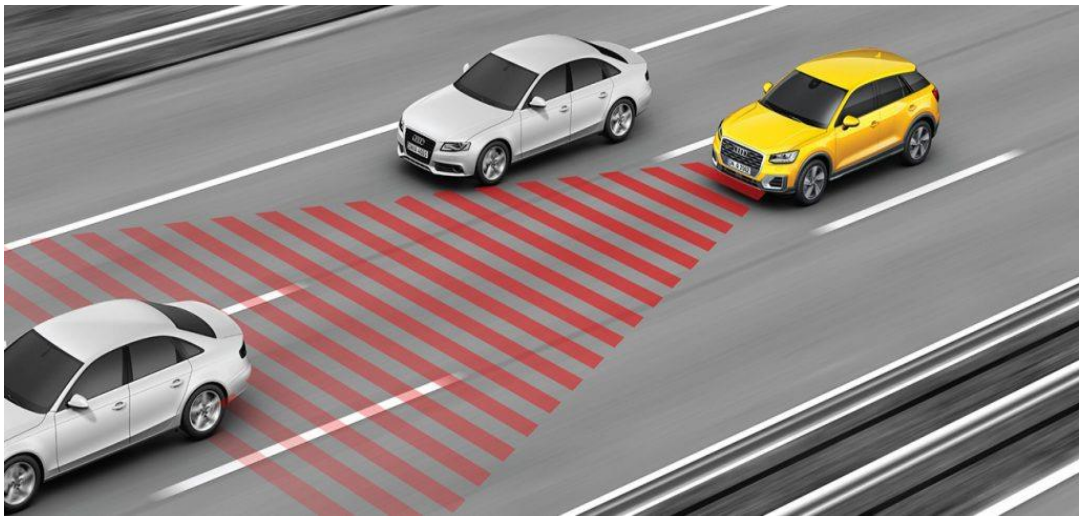


FIGURE1. 8: ADAPTIVE CRUISE CONTROL

In the latter chapters we will discuss and explain all the processes we did in order to get these features to be accomplished.

Chapter 2

Reverse engineering and improvement

1- What Is Reverse Engineering:

Reverse engineering, sometimes called back engineering, is a process in which programming, machines, and other products are deconstructed to extract design information from them. Often, reverse engineering involves deconstructing individual components of larger products. The reverse engineering process enables you to determine how a part was designed so that you can recreate it. Companies often use this approach when purchasing a replacement part from a piece of original equipment the creator.

2- How does reverse engineering work:

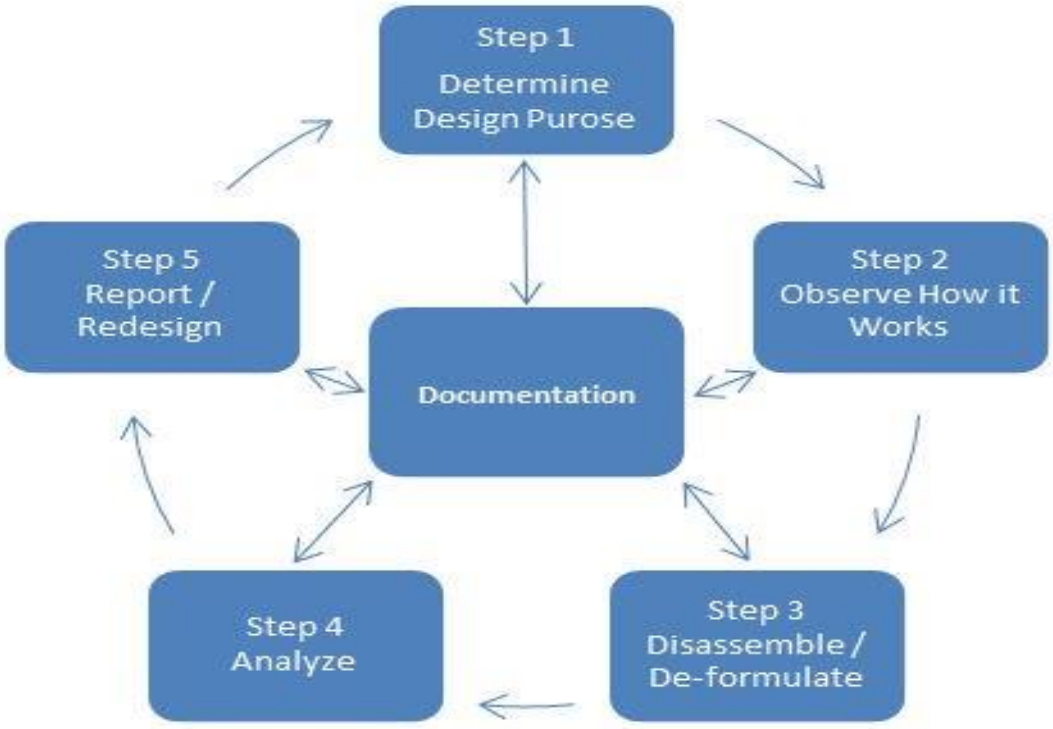


FIGURE2. 1: REVERSE ENGINEERING STEPS

Step 1: We have determined the purpose of the design for upgrade from normal ATV to autonomous vehicle.

Step 2: ATV was working by sample dc-circuit by a person on vehicle.

Step 3: We have disassembled ATV for upgrade.

Step 4: We have analyzed all the components and cost that we will need.

Step 5: This is redesign after upgrade.

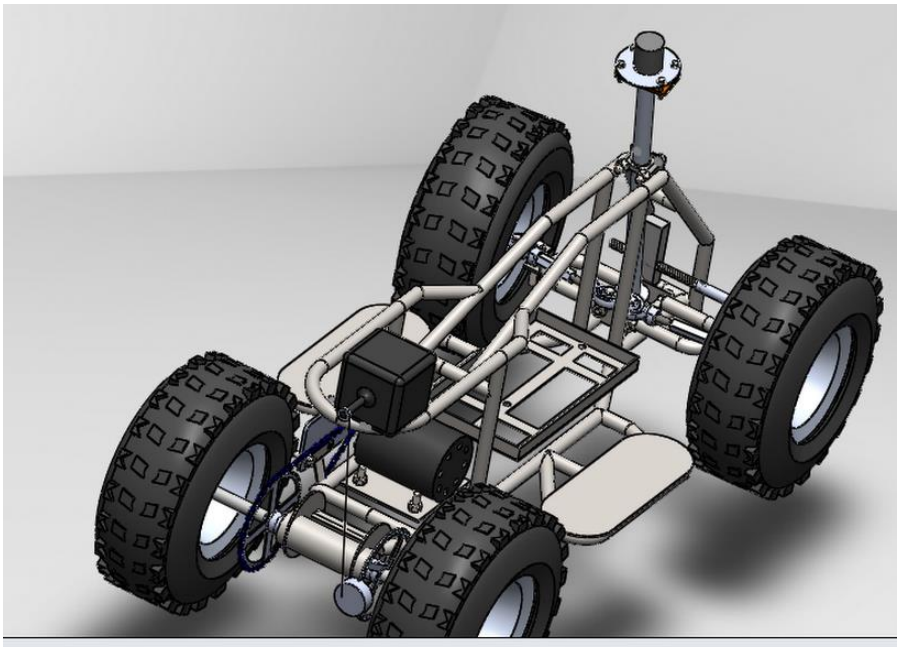


FIGURE2. 2 SOLID WORKS ATV

3- Initial specification of ATV:

- Length: 103 cm
- Width: 55 cm
- Height: 30 cm
- Wheel: 10cm*diameter 30 cm
- Thickness of tube 20mm*18mm
- Weight: 42 kg
- Maximum load weight: 55 kg
- Rated speed: 25 km/h
- Brake system: rear disc brake with mechanical control
- Motor watt: 800 watt
- Motor volt: 36 volt
- Rated torque: 2.8 N.M /rpm
- Maximum motor current: 30A ±1.5A

- Battery: 24V lead acid battery
- Run time: up to 40 min.
- Charging time: At least 6 hrs

4- Improvement specification:

We used a technology called drive-by-wire:

A technology known as drive-by-wire could change the way people drive. A car with this type of system would rely mainly on electronics to control a wide range of vehicle operations, including **acceleration**, **braking** and **steering**. Conventional cars mainly use hydraulic and mechanical technology to conduct these same basic vehicle operations, and although the systems are powerful, they can be overly complex, inefficient and conducive to wear and tear over the years.

In the following paragraphs, we will explain a simple idea of how we use this technology on the vehicle:

4.1- Software control of speed and acceleration:

We used the acceleration by wire technology to convert mechanical speed control to automatic control by the drive module to ccontrol of the motor input current and then to control the speed after feedback by the incremental encoder to obtain accurate speed.

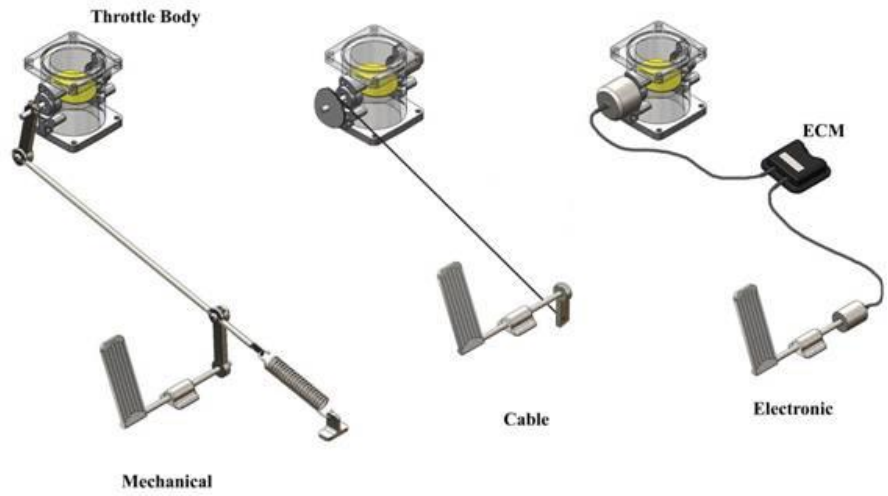


FIGURE2. 3: ACCELERATION BY WIRE

In subsequent chapters we will discuss and explain how this process is applied in detail.

4.2- Hardware control of speed:-

- motor drive for control on volt :-
  - Double BTS7960 H-bridge drive current.
  - Large current (43 A).
  - 5v power indicator on board.
  - Input supply voltage 5.5V to 27V.
  - Able to reverse the motor forward, two PWM input frequency up to 25kHZ.

- Encoder for log data of speed :-
  - 360 pluses/revelation (single-phase 360 p/r, two phase 4 frequency doubling to 1440 pulses).
  - Power source DC 5-24V
  - Shaft: 6*13 mm
  - Size: 38*35.5mm
  - Maximum mechanical speed 5000 r/min
  - Response frequency: 0-20 kHZ

4.3- Software control of brake:

We used braking-by-wire technology to replace mechanical brakes by hand to automatic control by a self-locking motor to pull the brake wire to stop the car until the limit switch to stop the brake motor for the following signal.

In next chapters we will discuss and explain how this process is applied in detail.

4.4- Hardware control of brake:

- Motor for brake by wire (self lock motor):
  - Motor with worm gear.
  - Power source DC 12V.
  - Shaft length 30 mm
  - Roller with wire.

- Limit switch:
  - MS.3
  - Dimensions 28*15.5*9.5mm
  - AC 125-250V -16A
  - DC 250V - 0.3A
  - DC 125V - 0.6A

4.5- Software control of steer angle:

We used steering-by-wire technology to replace mechanical steer by hand to automatic control by a linear actuator to control on the angle of steering and feedback control in this angle by incremental encoder.

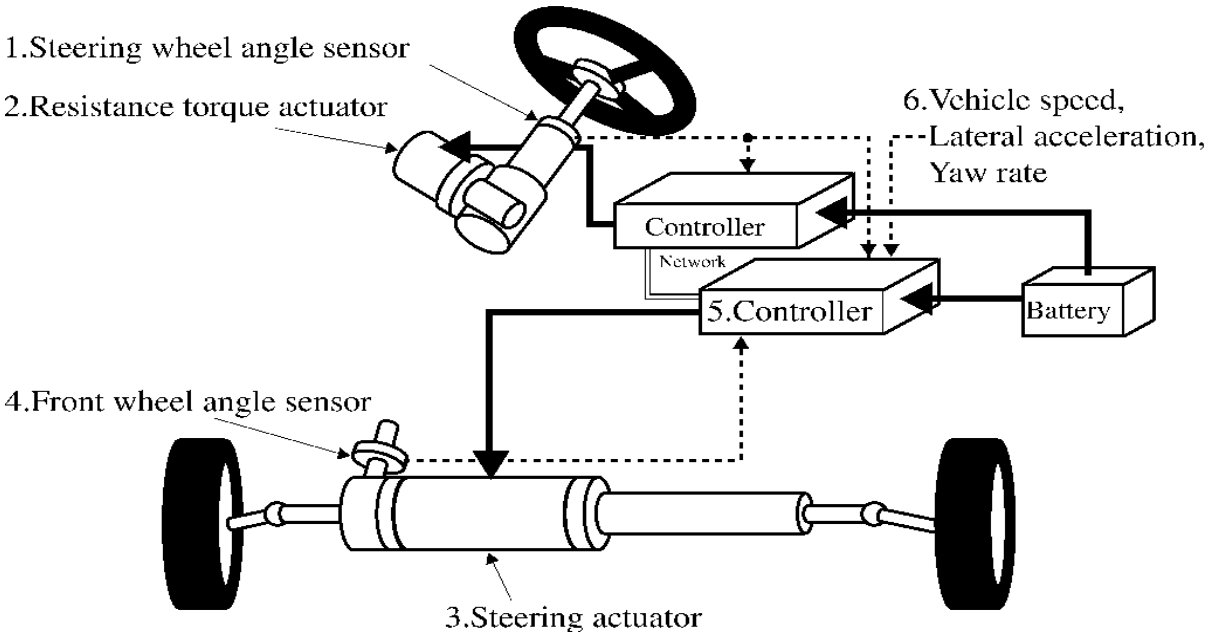


FIGURE2. 4: STEER ANGLE SOFTWARE

5- Why use linear actuator:

Electric Linear Actuator is a device that converts rotary motion of the electric motor to linear motion.

Linear actuator can be given:

1) Precise positioning. It can give us a very small angle of movement this gives high accuracy. Unlike ordinary motors, it will be difficult to control the angle with this precision

2) High efficiency. The lead screw is used as a power conversion part to achieve linear repeatable motion and reduce friction during movement; the power conversion efficiency can reach up to 80% – 90%.

3) A wide range of load capacity. Electric linear actuators load range from tens of kilograms to hundreds of tons.

4) Simplified structure, make full use of space. The electric linear actuators are a non-standard product and can be designed according to the allowed space.

In subsequent chapters we will discuss and explain how this process is applied in detail.

5.1- Hardware control of steer angle:

- Encoder for steering:
  - 2000 pluses/revelation.
  - Power source DC 5-24V
  - Shaft: 8*13 mm
  - Size: 40 mm*38mm
  - Maximum mechanical speed 5200 r/min

- Linear actuator :-
  - Lead screw 20 cm.
  - Lead screw diameter 12mm.
  - Power source DC 12V.

5.2- Installed sensor:

- Two increment encoder.
- LIDAR: measuring distance 8 m, measuring frequency 4 to 10 kHZ ,measuring angle 360 degree , baud rate 230400 .
- Camera: 180 degree.
- Jetson nano: Gpu 128-core Maxwell, Cpu quad-core arm a57, Memory 4GB, 4-Usb.
- Atmega 32**:** 32 Pin, 10 bit ADC, Max Operating Frequency 16MHZ.
- IMU 3_axis accelerometer, gyroscope, manometer.

Chapter 3

Mechanical Improvement

1- An Overview of X-By Wire Systems:

Nowadays many advanced technologies are incorporated with many disciplinary like mechanical, electrical, electronics, communication etc. Recent researches concentrate more on material reduction and safety features. This gives automotive industry a new concept called x-by wire systems in order to reduce some components with enhanced safety. The advancements in today's car give more comfortable to the driver and passenger. Embedded electronics, and more precisely embedded software, is a fast growing area, and software based systems are increasingly replacing the mechanical and/or hydraulic ones. The reasons for this evolution are technological as well as economical. "X" stands for the commanded action such as accelerating, braking and steering. By introducing x-by wire systems, large number of components will be replaced by electronic circuits and actuators.

2- Brake-By Wire system:

We replace the hand brake of the ATV at figure3.1 with motor to pull up the wire of the brake disc at figure3.2 to can control the brake from the microcontroller.

FIGURE3. 1: (HAND BRAKE)

FIGURE3. 2:(BRAKE DISC)

2.1- The mechanism on SOLIDWORK:

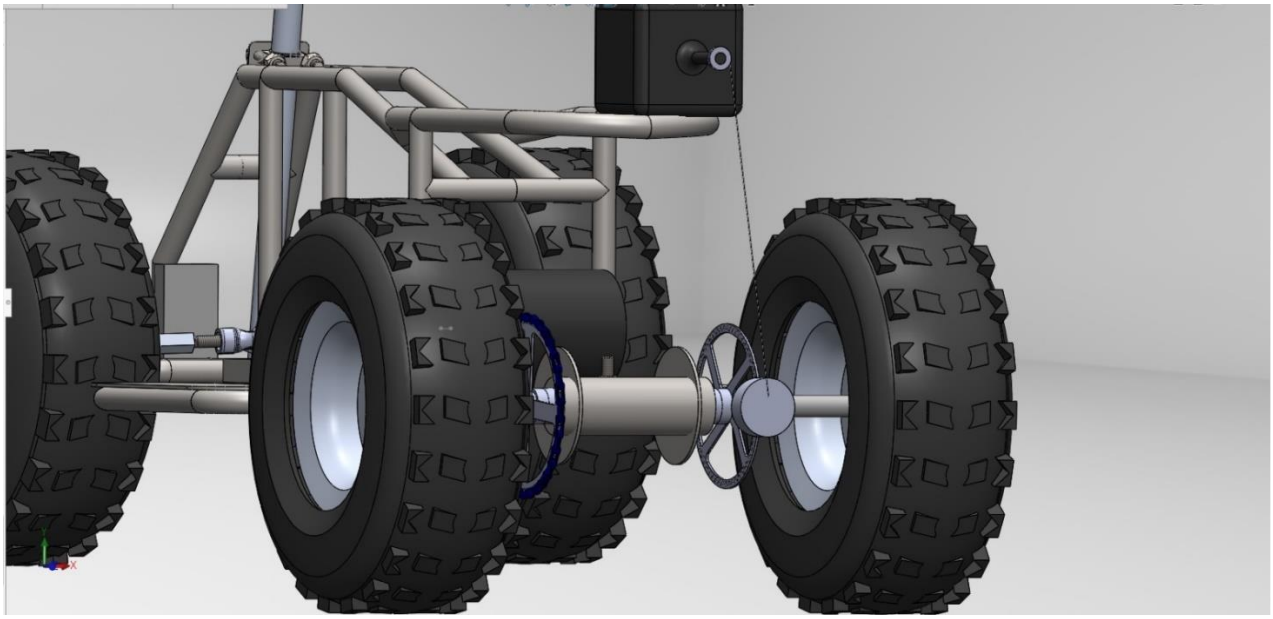


FIGURE3. 3:ATV MECHANISM

2.2- Components:

1. DC Motor with worm gear (Self lock)
2. Roller
3. Roller coupler

FIGURE3. 4: DC MOTOR

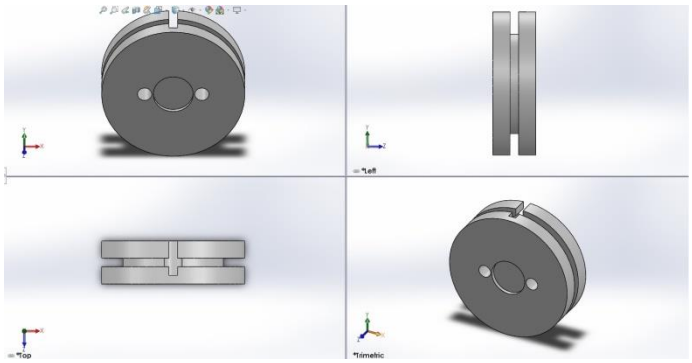


FIGURE3. 6: ROLLER 1

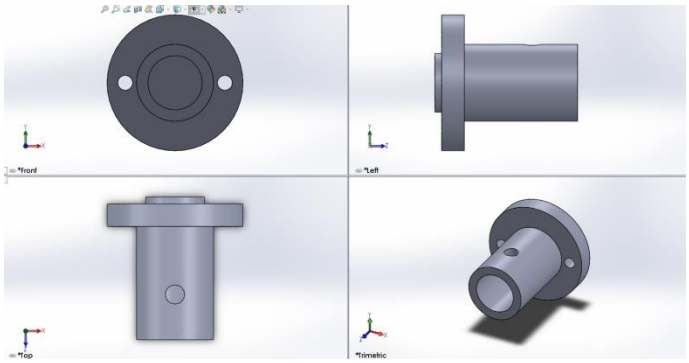


FIGURE3. 5: ROLLER COUPLER 1

2.3- Manufacturing processes:

We make the roller fig3.4 and the roller coupler fig3.5 at lathe. The material of the roller is artylon and the roller coupler from steel.

Why the material of the roller is artylon?

The artylon has high sliding strength, excellent tensile strength, high friction force and easy to operate on the lathe.

2.4- Why the material of the roller coupler is steel?

We need to make a hole thread to a nail, and we can't make it in the artelon so we choose steel to the roller coupler.

2.5- Motor specification:

1. Dc motor 12 volt.
2. Shaft diameter 12 mm
3. Motor weight 0.5 kg
4. Shaft length 30 mm

2.6- Roller specification:

1. Roller diameter 50 mm
2. Roller thickness 15 mm
3. The wire grove thickness 3 mm

2.7- Roller coupler specification:

1- Roller coupler diameter 25mm
2- Roller coupler length 26.5mm

- The working drawing and the all specification of the roller and the roller coupler at the appendix.

3- Steer-By wire system:

We make the linear actuator can change the angle of steering by change the angle of one wheel and the second wheel change his angle with the same value by the rod between the two wheel shown in figure3.6 the screw of the linear connect with the wheel by bearing with nuts and when the linear actuator move the wheel change his angle .

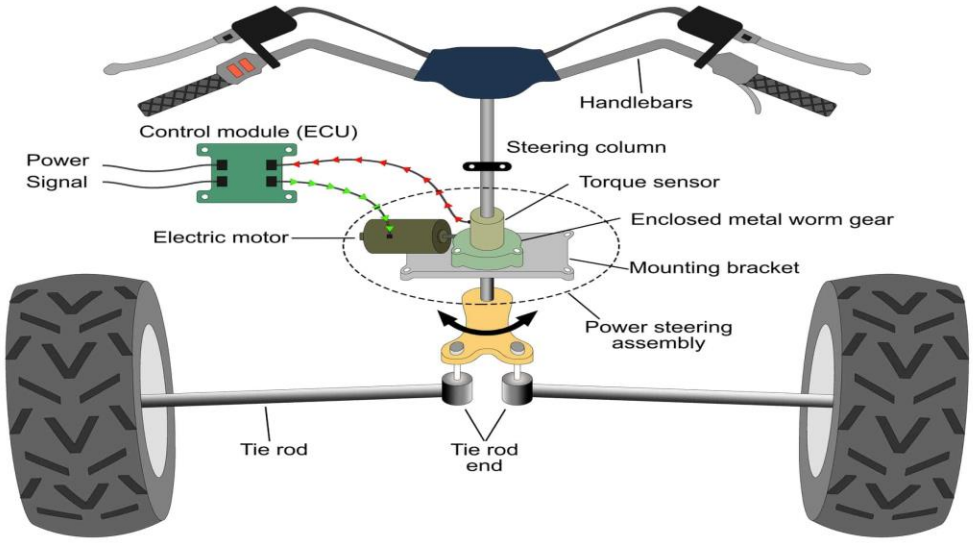


FIGURE3. 4:(THE STEERING MECHANISM)

3.1- Components

1. Linear actuator
2. End ball joint Bearing

FIGURE3. 5:(LINEAR ACTUATOR)

FIGURE3. 6: (END BALL BEARING)

3.2- Manufacturing processes:

We make housing to the linear actuator, we make plate from steel and hold it in the ATV body and hold the linear actuator at the plate collinear with the bearing in the wheel.

3.3- Linear Actuator specification:

1. Dc motor 12 v
2. Screw diameter 12 mm
3. Screw length 20 mm
4. Weight 0.5 kg
5. Housing length 8 cm
6. Housing width 5 cm
7. Screw pitch 2 mm

3.4- End ball bearing specification:

1. Inner diameter 10mm
2. Ball diameter 10 mm
3. Bearing length 25mm

The Mechanism CAD design:

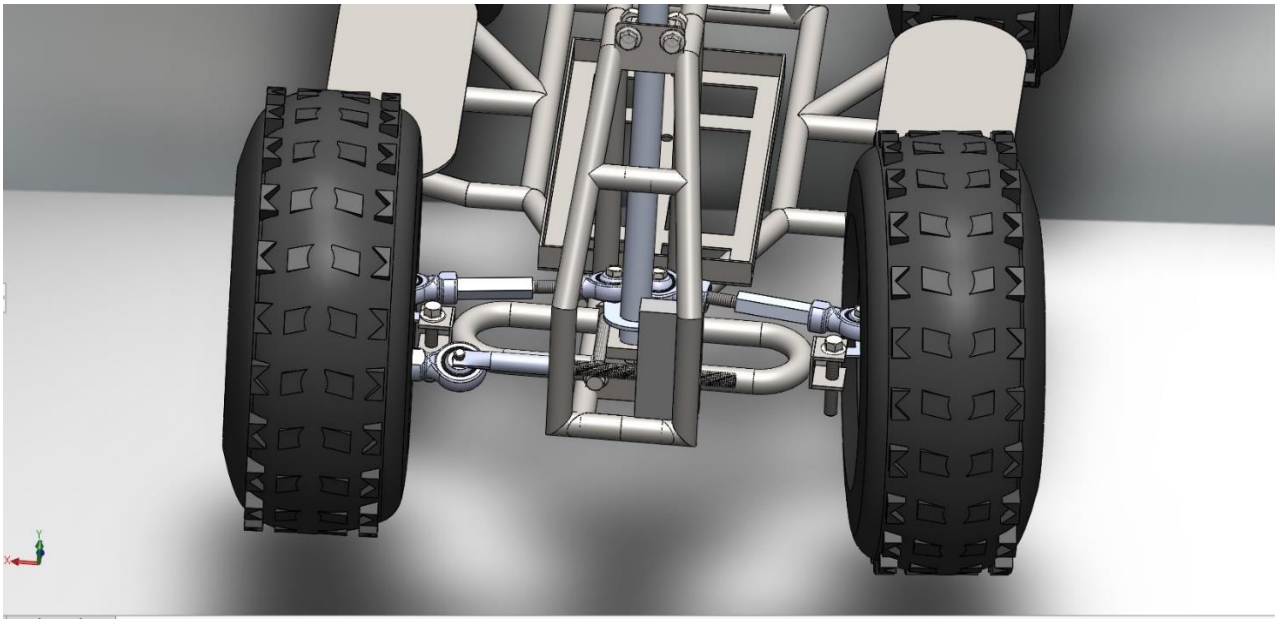


FIGURE3. 7:(STEERING MECHANISM)

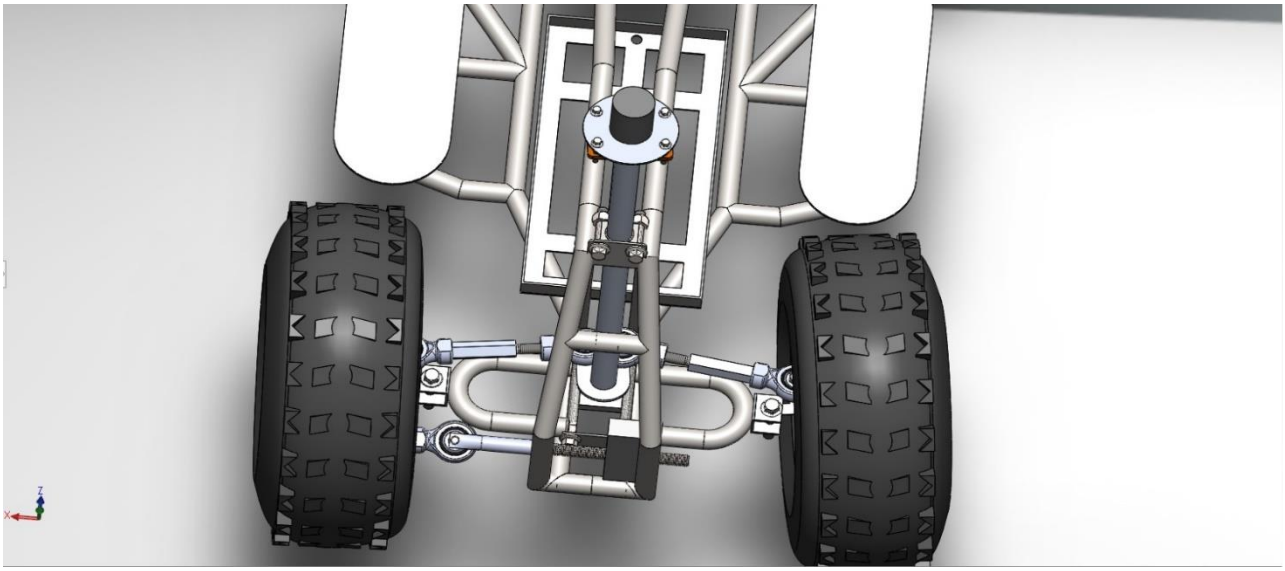


FIGURE3. 8: (STEERING MECHANISM)

4- ATV Dynamic Analysis

This section derives the driving power to ensure the ATV operation (Fig 3.11)

4.1- Road Loads

The road load consists of

$$\boldsymbol{F_w = F_{ro} + F_{ad} + f_{cr}}$$

- F_w = Road load
- F_{ro} = Rolling resistance force
- F_{ad} = Aerodynamic drag force
- f_{cr} = Climbing and downgrade resistance force

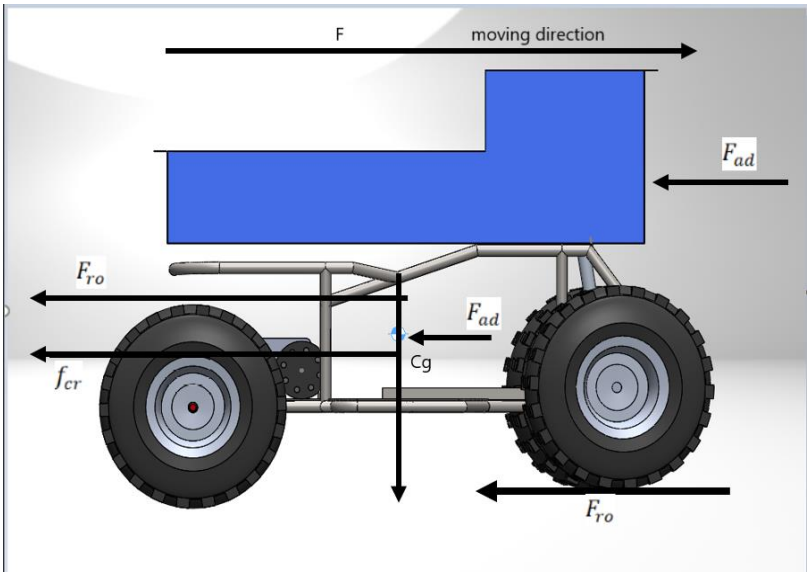


FIGURE3. 9 (ROAD LOADS)

The rolling resistance force F_{ro} is produced by the tire flattening at the roadway contact surface.

$$\boldsymbol{F_{ro} = \mu mg \cos \alpha}$$

- μ = Tire rolling resistance coefficient = 0.02 (Dry Asphalt)
- m = Payload = 52 kg
- α = Grade angle = 30°

$\boldsymbol{F_{ro}}$ = 0.02*52*9.81*$\boldsymbol{\cos 30}$ = 8.83 N

➢ The rolling resistance force can be minimized by keeping the tires as much inflated as possible.

Aerodynamic drag, *Fad*, is the viscous resistance of air acting upon the vehicle.

$$\boldsymbol{F_{ad}} = \frac{\mathbf{1}}{\mathbf{2}}\delta \boldsymbol{C_w A_f (V + V_o)^2}$$

- δ = Air density = 1.225
- C_w = drag coefficient = 0.3
- A_f = Front Area = 0.73*0.45 = 0.328 m^2
- V = ATV Speed = 25 Km/h = 6.9 m/s
- V_o = Wind Velocity = 90 Km/h = 25 m**/s**

$\boldsymbol{F_{ad}}$ = 0.5*1.225*0.3*0.328*(6.9+2$\mathbf{5}$)$^{\mathbf{2}}$ =61.33 N

The climbing resistance ($\boldsymbol{f_{cr}}$ with positive operational sign) and the downgrade force ($\boldsymbol{f_{cr}}$ with negative operational sign) is given by:

$$\boldsymbol{f_{cr}} = \pm \boldsymbol{mg \sin \alpha}$$

$\boldsymbol{f_{cr}}$ = 52*9.81* $\mathbf{sin\ 30}$ = 255.06 N

The following equation is derived due to the use of a reduction gear.

Maximum speed of the ATV is 25km\h = 6.9m\s
Wheel radius = 150mm

$\boldsymbol{\omega_{wheel}} = \frac{\boldsymbol{v}}{\boldsymbol{r}} = \frac{\mathbf{6.9}(\boldsymbol{m\backslash s})}{\mathbf{0.15}(\boldsymbol{m})}$ = 46 rad\s

$\boldsymbol{\omega_{wheel}} = \frac{\boldsymbol{\omega_m}}{\boldsymbol{i}}$ 46 = $\frac{\boldsymbol{\omega_m}}{\mathbf{5}}$ $\boldsymbol{\omega_m}$ = 230 rad/s

The load torque in the motor referential is given by.

$\boldsymbol{T_l} = \frac{\boldsymbol{R}}{\boldsymbol{i}}\boldsymbol{F_w}$ $\boldsymbol{T_l}$ = $\frac{\mathbf{0.15}}{\mathbf{5}}$ * (8.83+61.33+255.06)=9.756 N.m

5- Determination of Steering Torque

Steering Torque is defined as the force to turning the steering wheel. This can be calculated in either static condition i.e.
when the vehicle is stationary and in dynamic conditions.
Steering torque is maximum when the vehicle is stationary.

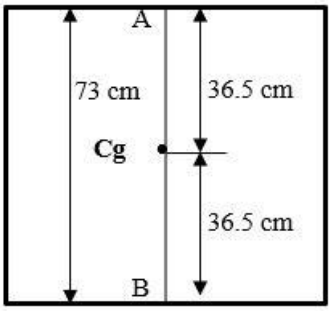


FIGURE3. 10: (CG FROM REAR AND FRONT)

Mass of ATV = 52 Kg
Mass acting on Center of Gravity = 52kg

Taking moment about B shown in Fig3.12

52*9.81*36.5 = Ra*73

Ra = 255.06 N

Reaction at each wheel

= (Ra\2 + mass of wheel)*9.81

= 255.06/2 + 4*9.81

=166.77 N

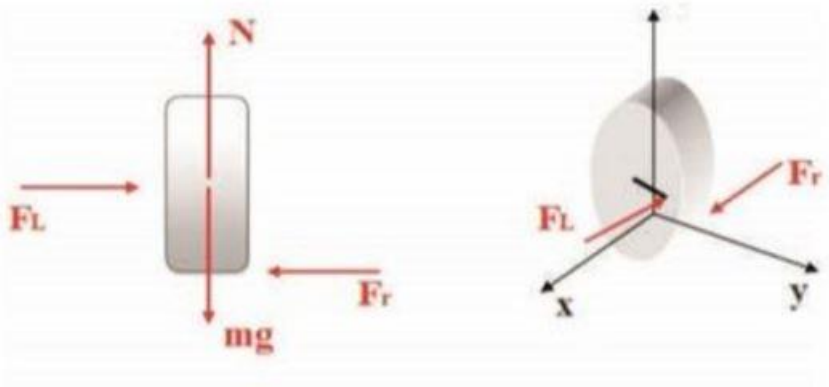


FIGURE3. 11: (FORCES ON TIRE)

When the ATV is stationary, the forces in all directions are in equilibrium (Fig 3.13)

$$\sum \boldsymbol{fx} = \mathbf{0} \; ; \boldsymbol{f_l} = \boldsymbol{f_r}$$

$$\sum \boldsymbol{fy} = \mathbf{0} \; ; \boldsymbol{N} = \boldsymbol{Mg}$$

- The tires used for the ATV are rubber tires and coefficient of friction μ = 0.02 (Dry Asphalt)

(data provided by various tire manufacturers)

The friction force at the tire is:

$= \mu * Reaction\ at\ each\ wheel$ **= 0.02 * 166.77 = 3.3354 N**
This torque should be equal or greater than the frictional resistance of the ground to be able to turn the tire.

From this, the value of steering arm torque is calculated by multiplying length of steering arm with the friction force FR. **Steering Arm torque = 3.3354*.25 = 0.833 N.m**

Chapter 4

Modeling and Control

1- DC-Motor modeling

1.1- DC-Motor Parameters:

A common actuator in control systems is the DC motor. It directly provides rotary motion and it can be easily connected to wheels, gears, etc. The dc-motor can be represented by a simple electrical circuit and a free body diagram of a rotor as shown in **Figure 1**.

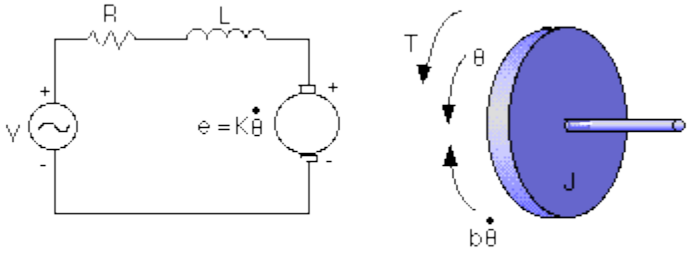


FIGURE4. 1: DC-MOTOR

In this model the input volt (**V**) and the output is the speed of the Shaft $\boldsymbol{\omega}$. By taking into consideration the all parameters of the motor like friction and other we can say that the psychical parameters of the motor are:

- **V:** voltage applied to the motor [volt]
- ***i*:** The current of the motor [Ampere]
- **θ**: The angel of the motor [Rad]
- $\dot{\theta}$**:** The speed of the motor [Rad/sec]
- **T:** Motor torque [N.m]
- **J:** Moment of inertia of the rotor [kg.m^2]
- **b:** Motor viscous friction constant [N.m.s]
- **Ke:** Electromotive force constant [V/rad/sec]
- **Kt**: Motor torque constant [N.m/Amp]
- **R:** Electric resistance [Ohm]
- **L:** Electric inductance [H]

1.2- DC-Motor Equations:

Generally, the Torque of the motor (**T**) is directly proportional with the Current of the motor and the strength of the magnetic field. since the magnetic field is constant the torque is proportional with the current only by a constant (***Kt***). The torque constant (***Kt***) is specific to motor's design, including its magnetic strength, number of wire turns, and armature length. The slope of the motor's torque-current curve is determined by the torque constant.

$$T = Kt * i \qquad (1)$$

The back emf is directly proportional to the speed of the motor shaft by a constant (***Ke***).

$$e = Ke * \dot{\theta} \qquad (2)$$

In SI units, the motor torque and back emf constants are equal, that is, $Ke = Kt$,we will use (***K***) to represent both the motor torque constant and the back emf constant.

By applying Kirchhoff's voltage law in the circuit.

$$L * \frac{di}{dt} + R * i = V - e \qquad (3)$$

By applying newton second law

$$T = J * \ddot{\theta} + b * \dot{\theta} \qquad (4)$$

By substation of equation (2) in (3)

$$L * \frac{di}{dt} + R * i = V - K * \dot{\theta} \qquad (5)$$

By substation of equation (1) in (4)

$$K * i = J * \ddot{\theta} + b * \dot{\theta} \qquad (6)$$

By applying Laplace transform to equation (5) & (6)

$$(L * s + R)i(s) = V(s) - k * s * \theta(s) \qquad (7)$$

$$k * i(s) = s(J * s + b * s)\theta(s) \qquad (8)$$

By subtitling $i(s)$ from equation (8) into equation (7)

$$\left(\frac{(L*s+R)(J*s+b*s)+K^2}{k}\right) s\theta(s) = V(s) \qquad (9)$$

$$\left(\frac{(L*s+R)(J*s+b*s)+K^2}{k}\right) = \frac{V(s)}{\dot{\theta}(s)} \qquad (10)$$

$$\frac{\dot{\theta}(s)}{V(s)} = \frac{k}{(L*s+R)(J*s+b*s)+K^2} \qquad (11)\ \text{SPEED VOLT EQUATION}$$

$$\frac{\theta(s)}{V(s)} = \frac{k}{s*((L*s+R)(J*s+b*s)+K^2)} \qquad (12)\ \text{POSITION VOLT EQUATION}$$

1.3- Parameters acquiring:

In normal cases most of the parameters can be acquired form the motor datasheet but since the motor we are working with is of an unknown origin and the datasheet could not be acquired so other methods are used to acquire those parameters the method that we use is curve fitting method .

Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points , those series of data is acquired by applying an known input to the system (volt) and recording the output of the system (speed or angle). Curve fitting can involve either interpolation, where an exact fit to the data is required, or smoothing, in which a "smooth" function is constructed that approximately fits the data.

This process is done using the MATLAB Simulink parameter estimation tool were the input (volt) and the output (speed) is applied then a series of iteration is done to find the best values of the parameters to fit the output as seen in figure 4-2 and 4-3.

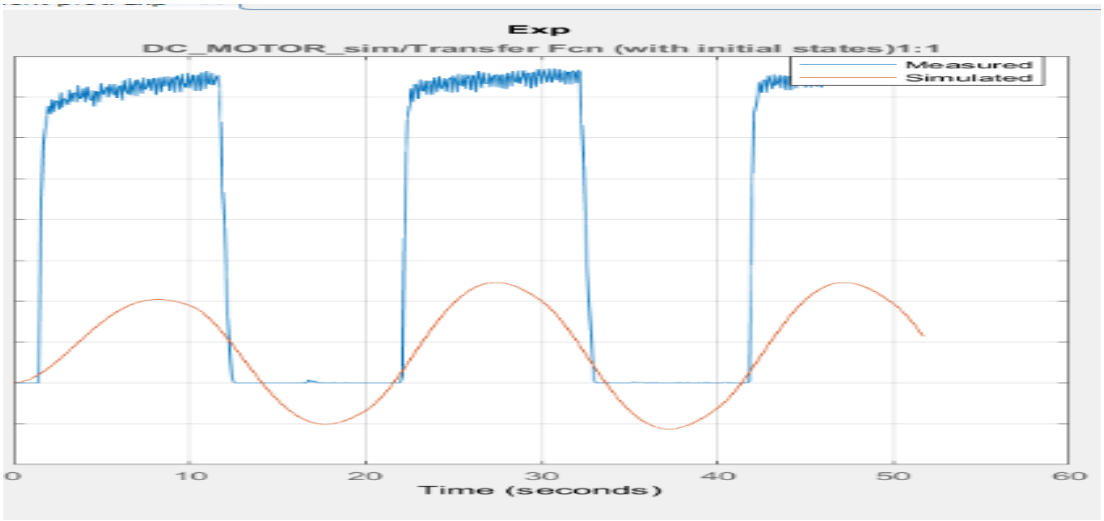


FIGURE4. 2: DC MOTOR TRANSFER FUNCTION

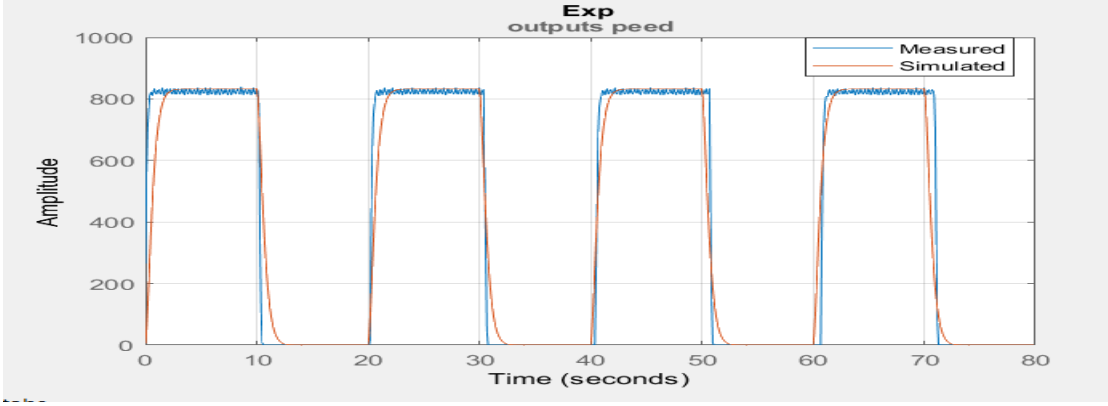


FIGURE4. 3: OUTPUT RESPONSE

The result of these operation is that we were able to acquire the parameters of the dc motor without the datasheet but by logging the input and the output of the motor and finding the best parameters that can make the model respond similarly.

2- Feedback and Control

2.1- Why use a feedback control:

in the last part we found a relation between the speed and the volt we set a specific volt to get a speed but the is no way of knowing if the motor has reached the required speed that is called an **Open loop system** where the output of the model does not affect the input.

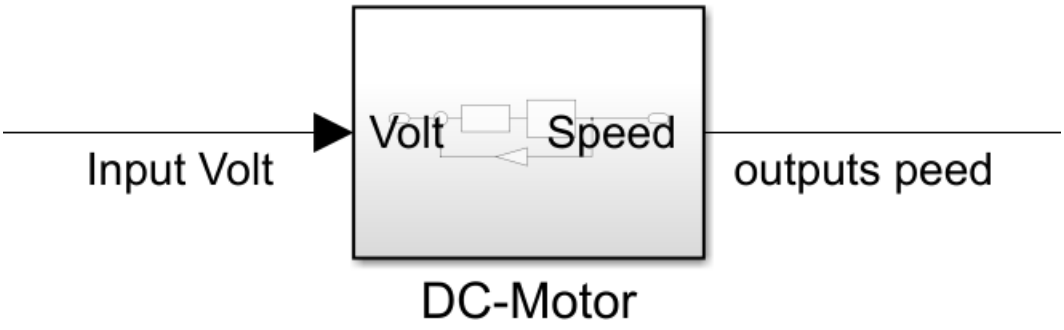


FIGURE4. 4: OPEN LOOP SYSTEM

So, by adding any disturbance to the system the output well change therefore we use a feedback from the output to be affect the inputs and that is called a **Close loop system.**

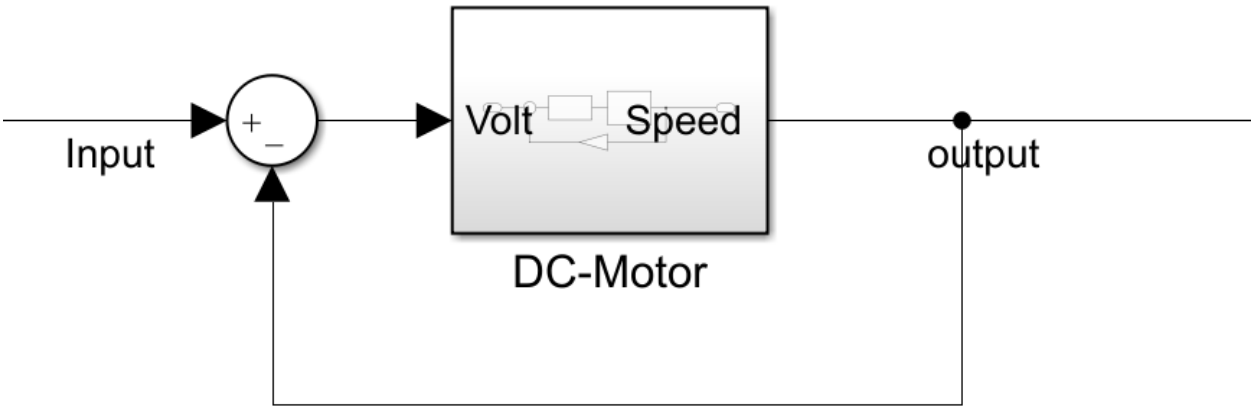


FIGURE4. 5: CLOSE LOOP SYSTEM

What does the feedback do? By which a measure of the real output of the system that includes the effect of the disturbances and compare it by the input. The comparison between the input (the demand value) and the output (the obtained value) give the system error that depends on all the system imperfection effects including the disturbances. The error signal can be obtained by means of mathematical subtraction that normally obtained by a device called comparator.
The feedback will not solve the problem interlay there will be a SteadyState error

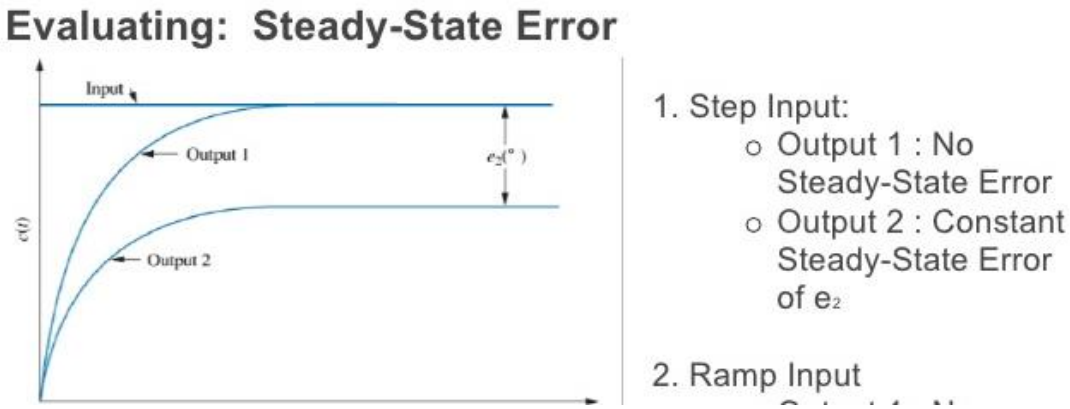


FIGURE4. 6: STEADY STATE ERROR

So, know we have the steady state problem and we have another problem that in this current application we want to be able to insert a required speed not a volt. The beat solution to these problems is to use a controller and the controller we will be using is an PI controller.

2.2- PID Controller:

Proportional–integral–derivative controller (**PID controller**) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an **error value** as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted *P*, *I*, and *D* respectively) as shown in figure 4-6.

let first disscus each term of the controller alone.

P- Controller:

Proportional or P- controller gives output which is proportional to current error e (t). It compares desired or set point with actual value or feedback process value. The resulting error is multiplied with proportional constant to get the output. If the error value is zero, then this controller output is zero.
This controller never reaches the steady state condition. It provides stable operation but always maintains the steady state error.

PI- Controller:

Due to limitation of p-controller where there always exists an offset between the process variable and set point, I-controller is needed, which provides necessary action to eliminate the steady state error. It integrates the error over a period of time until error value reaches to zero. It holds the value to final control device at which error becomes zero.
Integral control decreases its output when negative error takes place. It limits the speed of response and affects stability of the system. Speed of the response is increased by decreasing integral gain Ki.

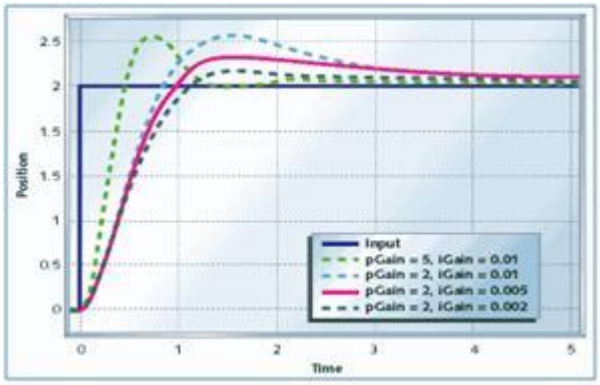


FIGURE4. 7: PI-CONTROLLER RESPONSE

In figure 4-5, as the gain of the I-controller decreases, steady state error also goes on decreasing. For most of the cases, PI controller is used particularly where high speed response is not required.

While using the PI controller, I-controller output is limited to somewhat range to overcome the integral wind up conditions where integral output goes on increasing even at zero error state, due to nonlinearities in the plant.

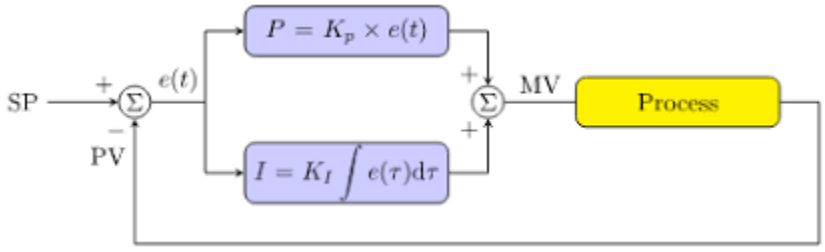


FIGURE4. 8: PI-CONTROLLER

PID- Controller:

I-controller doesn't have the capability to predict the future behavior of error. So, it reacts normally once the set point is changed. D-controller overcomes this problem by anticipating future behavior of the error. Its output depends on rate of change of error with respect to time, multiplied by derivative constant. It gives the kick start for the output thereby increasing system response.

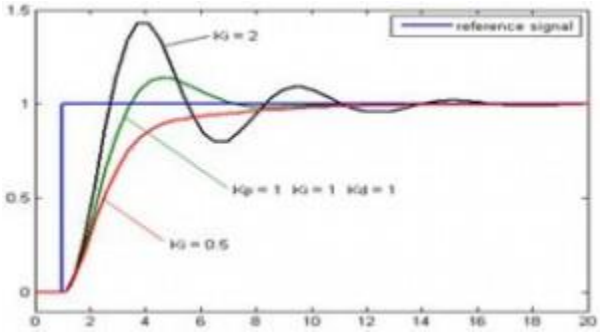


FIGURE4. 9: PID-CONTROLLER RESPONSE

In the above figure response of D controller is more, compared to PI controller and also settling time of output is decreased. It improves the stability of system. Increasing the derivative gain increases speed of response.
But the major drawback of the D-controller is that any spike in the data or bid disturbance will make the controller behave badly so we need a filter on the derivative term to make it behave normally.

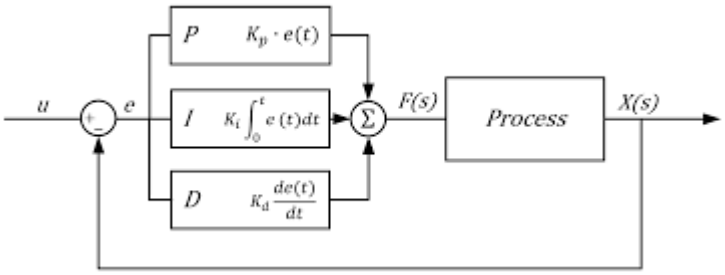


FIGURE4. 10: PID-CONTROLLER

2.3- Choosing a controller:

Since the loaded motor response is slow so will be applying a **PI-Controller** and the equation representing the controller is

$$e(t) = setpoint - processvalue \quad (13)$$

$$u(t) = kp * e(t) + ki * \int e(t)\, dt \quad (14)$$

But since the motor is being controller by a discrete system it is needed to transfer this function to the z-domain

$$U(z) = kp * E(z) + ki * \frac{z+1}{z-1} * Ts * E(s) \quad (15)$$

Where

Ts: is the sampling time

2.4- Tuning the controller:

Tuning is the process in which we a acquire the values of kp and ki there are many different way and tools by which we can acquire those variables like trial and error but since we have a model of the dc motor we can easily get the variables using MATLAB SIMULINK .

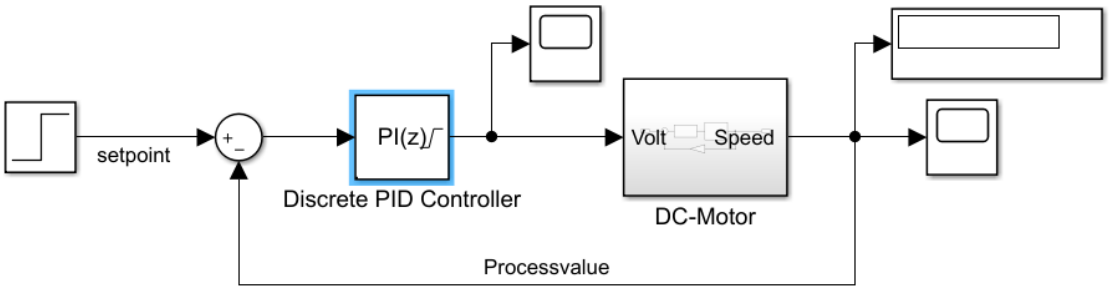


FIGURE4. 11: PI-CONTROLLER USING MATLAB SIMULINK

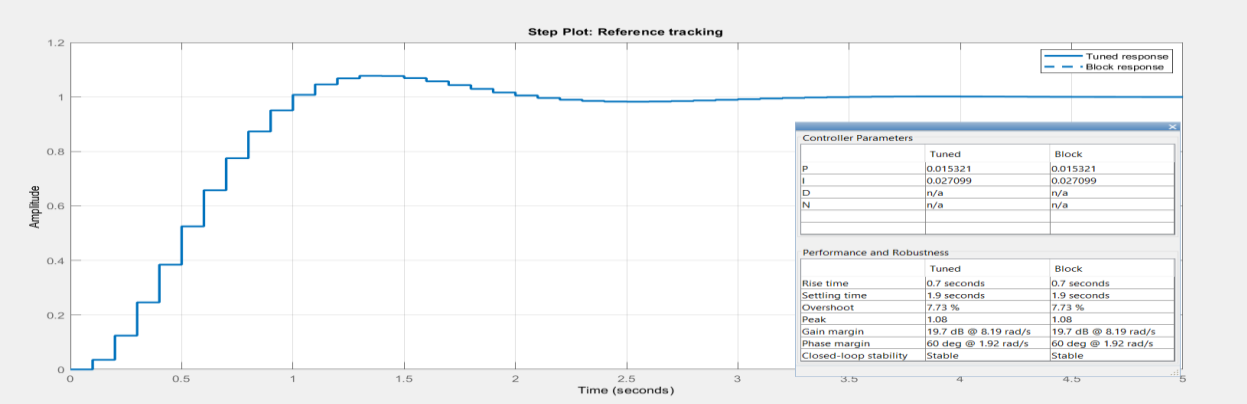


FIGURE4. 12: SYSTEM RESPONSE

By this tool we can easily change the Parameters of system like Overshoot, settling time.

2.5- PI-Controller Implementation:

Since we are using a Microcontroller to control the motor, we must be able to implement the PI-control theory to Code to be used on the microcontroller, so we need to change to Z-domain equation into a code

$$U(z) = kp * E(z) + ki * \frac{z+1}{z-1} * Ts * E(s)$$

Will be change to:

$$\boldsymbol{U(t) = u(t-1) + kp * e(t) + ki * (e(t) + e(t-Ts)) * Ts * e(t)} \quad \textbf{(16)}$$

2.6- Conclusion:

By acquire a relation between the motor input (volt) and the output (speed or angle) and get all the parameters we can construct a controller to be able set a required speed or angle and overcome any disturbance.

Chapter 5

Embedded System

1-Embedded System

An embedded system is a combination of computer hardware and software integrated to perform a particular function. It uses a Microcontroller or Microprocessor to perform a one job. It is a device that has no operating system.

Embedded System hardware:

Embedded system hardware type is microprocessor or microcontroller based.
Microprocessors are visually difficult to tell apart from microcontrollers, but while microprocessors can only implements a central processing unit, so it needs the addition of other components such as memory chips.

Microcontrollers design is considered a self-contained systems, and doesn't include a CPU only, but also memory and peripherals such as flash memory, RAM or serial communication ports.

2-History of Embedded System:

Embedded systems date back to the 1960s. Charles Stark Draper had made an improved integrated circuit (IC) in 1961 to reduce the size and weight of the "Apollo Guidance Computer", the digital system installed on the Apollo Command Module and Lunar Module. The first computer used ICs has helped astronauts to collect real-time flight data. In 1965, the D-17B was developed, that computer is used in the Minuteman I missile guidance system. The D-17B is hugely known as the first mass-produced embedded system. the first embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system. By the late 1960s and early 1970s, The first microcontroller was developed by Texas Instruments in 1971. The TMS 1000 series, which became commercially available in 1974, contained a 4-bit processor, read-only memory (ROM) and random-access memory (RAM), and cost around $2 apiece in bulk orders.
In 1987, the first embedded operating system, was released by Wind River.

3-What is Microcontrollers?

Microcontroller is considered as a single integrated circuit computer present which is devoted to perform one task and execute a specific application. Microcontroller contains memory, programmable input/output peripherals and also a processor.

AVR Microcontroller :

AVR refers to an Advanced Virtual Reduced Instruction Set Computer (RISC), it is a custom-made Harvard construction 8 bit RISC isolated chip micro-controller. This microcontroller was invented in the year 1966 by Atmel.

Classification of AVR Controllers:

- TinyAVR : Smaller size, less memory, only suited to simpler applications
- MegaAVR : These are the most common and used controllers, they have good amount of memory (reach to 256 KB), high number of internal peripherals and suited to moderate complex applications
- XmegaAVR : These are used for commerciall for complex applications, which require huge program memory and high speed

4-Why ATmega32 Not Arduino

We use ATmega32 because it has higher speed and higher performance than arduino. ATmega32 is a microcontroller based on the AVR improved RISC construction and is a low-power CMOS 8-bit. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughputs near to 1 MIPS per MHz letting the system designer to optimize power consumption versus processing speed.

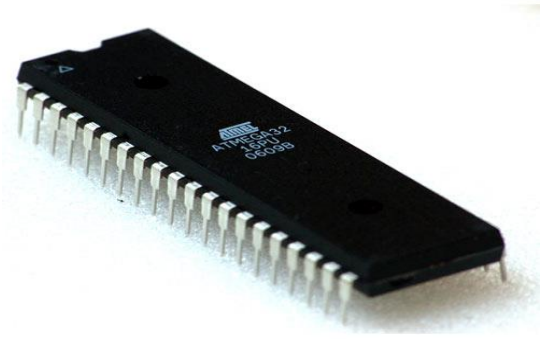


FIGURE5. 1: ATMEGA32

Features:

• High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
• Advanced RISC Architecture
  – 131 Powerful Instructions
  – Most Single-clock Cycle Execution
  – Registers are 32 × 8 General Purpose
  – Fully Static Operation
  – Reach to 16 MIPS Throughput at 16MHz
• High Endurance Non-volatile Memory segments
  – 32Kbytes of In-System Self-programmable Flash program memory
  – 1024Bytes EEPROM
  – 2Kbytes Internal SRAM
  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  – Data keeping for 20 years at 85°C/100 years at 25°C(1)
• Peripheral Features
  – Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  – One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  – Has a Real Time Counter added with Separate Oscillator
  – Four PWM Channels
  – 8-channel, 10-bit ADC
  – Byte-oriented Two-wire Serial Interface
  – Programmable Serial USART
  – Master/Slave SPI Serial Interface
  – Is a Programmable Watchdog Timer with Separate On-chip Oscillator
•Internal Calibrated RC Oscillator
•External and Internal Interrupt Sources
•Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby Extended Standby
• I/O and Packages
  – 32 Programmable I/O Lines
  – 40-pin PDIP

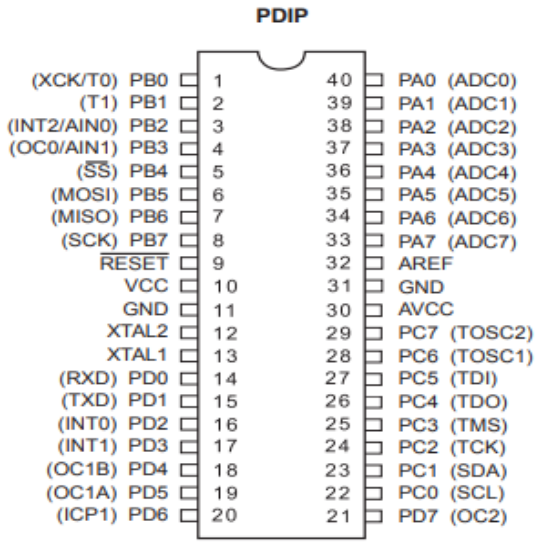


FIGURE5. 2: PINOUT ATMEGA32

• Operating Voltages
  – 4.5V - 5.5V
• Speed Grades
  – 0 - 16MHz for ATmega32
• Power Consumption at 1MHz, 3V, 25°C
  – Active: 1.1mA
  – Idle Mode: 0.35mA
  – Power-down Mode: < 1µA

5- Microcontroller abstraction layer:

1. DIO (Digital Input Output)
2. Interrupt
3. Timer
4. PWM(pulse width modulation)
5. UART

5.1- DIO (Digital Input Output)

A Digital Input Output is a peripheral that deals with digital signals, either by generating a digital signal (Output Mode) or by receiving it (Input Mode).

In our project we use microcontroller AVR Atmega32. It has 32 DIO pins grouped as following:

1. PORTA has 8 DIO pins from A0 to A7
2. PORTB has 8 DIO pins from B0 to B8
3. PORTC has 8 DIO pins from C0 to C8
4. PORTD has 8 DIO pins from D0 to D8

Each pin can work either in input mode or output mode.

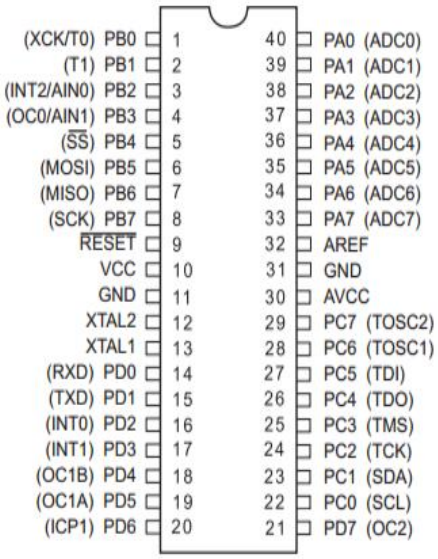


FIGURE5. 3: ATMEGA32 PINS

Every port has 3 control registers

i. DDR
ii. PORT
iii. PIN

The size of each register is 8 bit, every bit corresponding to 1 pin of the port

i. 1-**DDR** (data direction register): in this register we can define the pin is output or input
ii. 2-**PORT**: this register is used in output mode to set the digital output value
iii. 3- **PIN**: we use this register in case the pin is defined input

5.2- Interrupt

Interrupt is an event that disturbs the normal execution sequence of the code and makes the processor to jump to another piece of code to execute called Interrupt Service Routine (ISR). After the processor finish executing of the ISR, it returns back to the interrupted code to continue from the interrupted instruction**.**

FIGURE5. 4: INTERRUPT.

5.3- Timer and counter

What is counter?

Simply counter is a peripheral that counts events in a specific register. These events are electrical signals like rising edge or falling edge. If these events are periodic (Comes every defined period) then this counter is counting time, hence it is called Timer.

Timer:

We can say that a timer is a normal counter but counting a periodic signal. So that we can calculate the time from the value in the timer register.

In summary, the timer and the counter are the same peripheral, depending on the input we could classify if it is working as timer or as counter

Timer terminology

Resolution: Number of bits represents the timer register

Timer Tick Time: The time between two consecutive events,

I.e. time needed to increment the timer register by one

. It shall be equals to $\frac{\textbf{prescale}}{\textbf{frequency}}$

Timer Count: Number of counts needed by the timer to count from 0 till it reaches 0 again. It shall be equals to $\mathbf{2^{Resolution}}$

Timer Overflow: The time needed by the timer to count from 0 until it reaches 0 again. It shall be equals to **Timer Count x Timer Tick Time**

Prescaller: Prescaller is a division factor for the system clock (Processor Clock) before it is applied to the timer.

Types of timer:

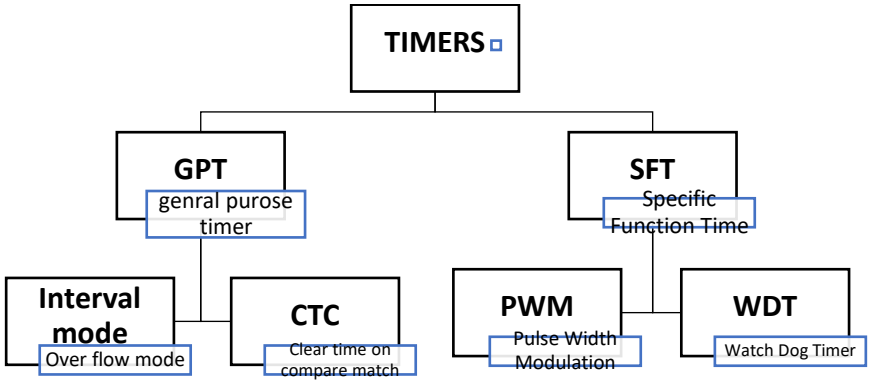


FIGURE5. 5: TIMER TYPES

In ATMEGA32, we have three different kinds of timers:

TIMER0

- 8-bit timer
- CTC mode
- Phase Correct PWM
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources

TIMER1

- 16-bit timer
- Two Independent Output Compare Units
- One Input Capture Unit
- Phase Correct (PWM)
- Variable PWM Period
- Four Independent Interrupt Source

TIMER2

- 8-bit timer
- CTC mode
- Phase Correct PWM
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources
- Allows clocking from External 32 kHz

5.4- Pulse Width Modulation:

- Pulse Width Modulation (PWM) is a method for changing how long a square wave stays “on”.
- The on-off behavior changes the average power of the signal.
- If signal toggles between on and off quicker than the load, then the load is not affected by the toggling.

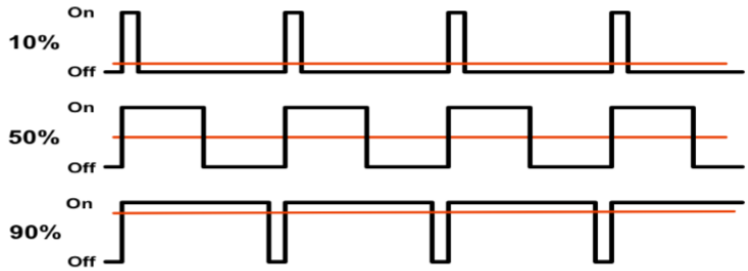


FIGURE5. 6: PULSE WIDTH MODULATION.

WHY PWM?

Because the PWM is on for a period and off for another period, the total power is a part for the maximum power (The On Power). It looks a like a potentiometer but with some differences.

TABLE 1

PWM	Potentiometer
• Automatic control • The total power consumed in the load	• Manual control • Part of power consumed at the potentiometer

PWM parameters

Amplitude: The voltage difference between the on state and the off state.

Period: The repetition time for PWM. Period = On_Time + Off_Time

Duty cycle: The percentage of the On_Time over the total time

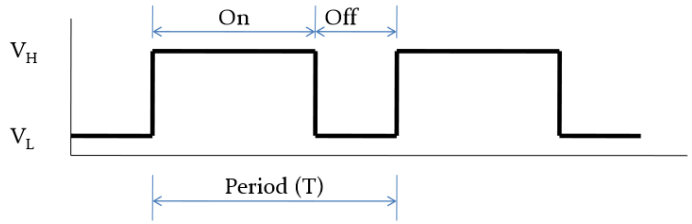


FIGURE5. 7: PWM PERIOD.

The duty cycle is a percentage measurement of how long the signal stays on.

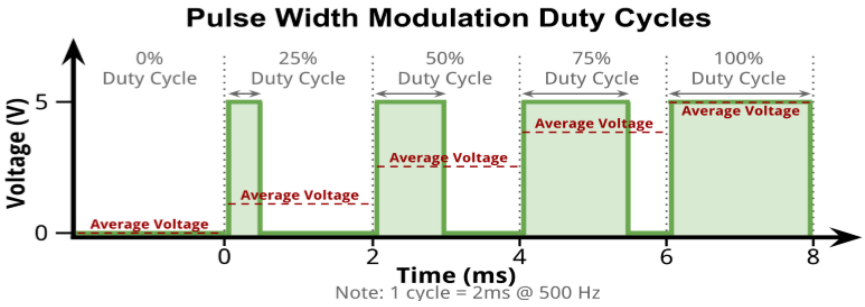


FIGURE5. 8: PWM DUTY CYCLES

PWM important equations

Frequency = waves per second

Period =T_{total}= time on + time off= seconds per wave = 1/frequency

Duty cycle (D) =time on/ (time on + time off)

Average signal can be found as: V_{avg}=D.V_H+ (1-D).VL

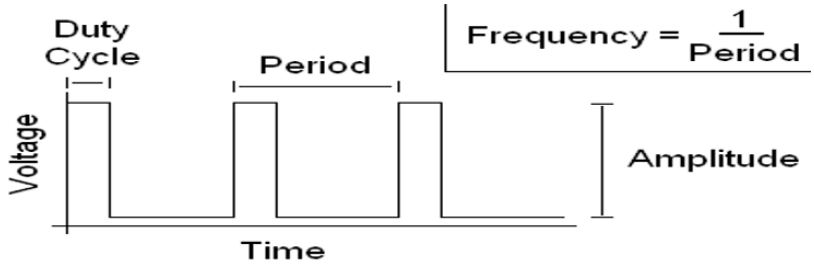


FIGURE5. 9: PWM PARAMETERS.

5.5- UART (Universal Asynchronous Receiver Transmitter)

It is a serial communication protocol that consists of one wire for transmitting data and one wire to receive data.

A common parameter is the baud rate known as "bps" which stands for bits per second. If a transmitter is configured with 9600bps, then the receiver must be listening on the other end at the same speed.

UART hardware interface

Each node has a line called Tx (Transmission line) and another one called Rx (Receive Line). The Tx of one node shall be connected to Rx of the other node and vice versa.

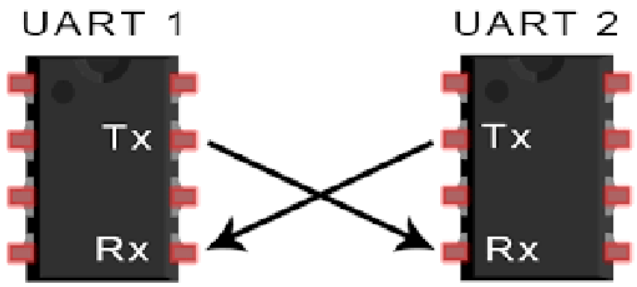


FIGURE5. 10: UART

UART specifications:

- Weird :connection by wire
- Serial: sent data in serial bit by bit
- Full duplex: can sent and receive at same time
- Asynchronous :doesn't need clock , agree on speed of transmission and receiving before transfer data
- Peer to peer: the two nodes have the same level of importance

6-DC Motor Speed

1- Read the motor speed using an incremental encoder
2- Control the motor speed
3- Connect the motor to the microcontroller using BTS7960 **driver**

6.1- Read motor speed

Steps:
1. Implement MCAL and HAL drivers
2. Determine pulses per revolution (PPR)
3. Use pulses to read the motor speed
4. Simulate C-code with circuit in Proteus

Encoder specifications:

- 360 pulse per revelation (PPR) encoder

Explanation of the code:

We use a digital external interrupt (DEI) to count the pulses of encoder by generate an interrupt every each falling age, and in every interrupt the counter increments by one.

Then we use the timer in overflow mode to read the number of pulses counted in the DEI every 0.1 second.
As in this mode every time the timer register overflow in gives an interrupt and we can count the number of interrupts for the needed interval of time by this equations.

$$timer\ tick = \left(\frac{\text{prescale}}{frequency}\right)$$
$$timer(overflow) = timer\ tick * 2^{resolution}$$
$$counter\ value\ = \left(\frac{\text{needed time}}{timer(overflow)}\right)$$

Then calculating the speed in RPM using this equation:

$$speed\ in\ (rpm) = \left(\frac{\frac{no.of\ pulses}{\text{time interval}}}{pulses\ per\ revolution}\right) * 60$$

6.2- Control the motor speed

Steps:

1) Implement MCAL and HAL drivers
2) Control speed using timer PWM mode
3) Simulate C-code with circuit in Proteus

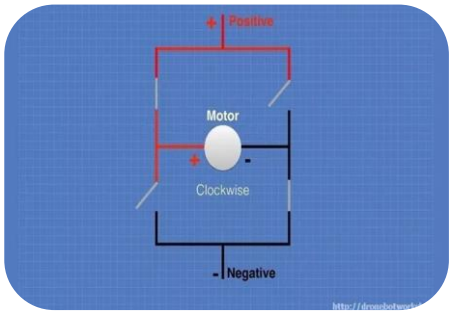
Code explanation:

First of all implement the driver for timer PWM mode which allows us to on for a period and off for another period, the total power is a part for the maximum power (The On Power).
Simply it allows us to control the amount of volt passes to the motor driver, by control a specific register to make the compare match between TCNT (register) and OCR (register), and cleared at TOP.
So that we can control the amount of voltage pass by putting number from 0 to 255 for an 8 bit register in OCR (register) ,that relative to the voltage from 0 to 5 volt

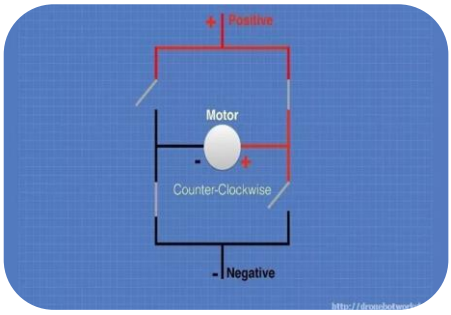
6.3- Connect the motor to the microcontroller using BTS7960 driver

Why we need a driver?

- As motor needs a high power and the microcontroller generates 5V as a maximum volt
- To control the motor rotation clockwise(CW) and counterclockwise (CCW)
- To prevent the back EMF not to burn pins in the microcontroller



if we close (i.e. turn on) two of the switches you can see how the voltage is applied to the motor, causing it to turn clockwise.



Now we'll open those switches and close the other two. As you can see this causes the polarity of the voltage applied to the motor to be reversed, resulting in our motor spinning counterclockwise.

7- Steering Angle

After we have calculated the speed through the encoder previously , we calcualte the angular position or steering angle using incremental rotary encoder.
- We use 2000 pulse per revolution (PPR) incremental encoder because we need high resolution and to measure the angle precisely.
- The measured angle encoders determine the rotational position of a load in relation to a shaft or point. Encoder's angle provides output that matches to displacement and the reading device processes that data into angular readings.
- An encoder with one set of pulses would not be useful because it could not indicate the direction of rotation, so we use two tracks with sectors positioned 90 deg out of phase like figure 4 shows.

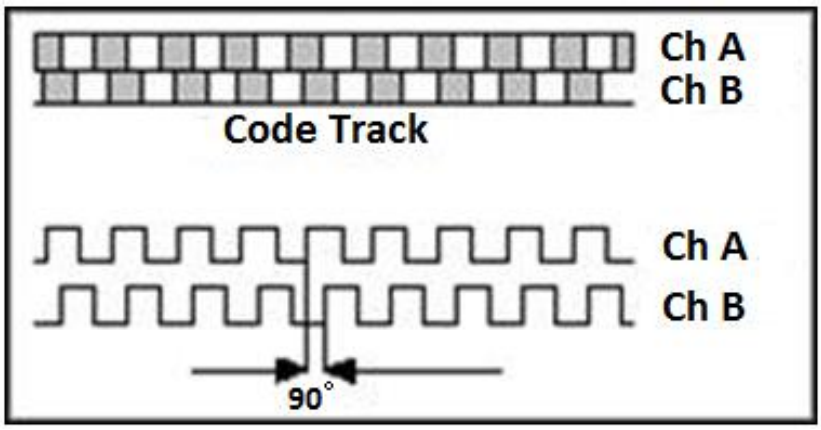


FIGURE5. 11: QUADRATURE

Encoder A and B Output Signals

These two output channels of the quadrature encoder specify the position and direction of rotation.
- Lets say for example if A leads B, the disk is rotating in a clockwise (CW) direction. If B leads A, then the disk is rotating in a counter-clockwise (CCW) direction.
- We need to consider is how those values are converted to position. Depending on the types of the decoding used the edge counts are converted to position through the process.
- There are three basic types of decoding, X1, X2, and X4, here we are going to use X4 decoding.
- At first we used X1 decoding but it wasn't the best solution we are looking for in our project.
- X4 decoding : Figure 4 Shows the quadrature cycle and the resulting increments and decrements happening in each edge of channle A and B for X4 decoding.

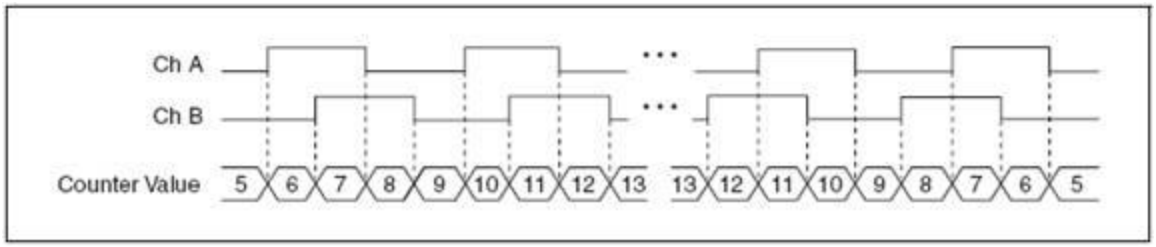


FIGURE5. 12: X4 ENCODING

- By counting the rising edges of channel A and channel B and the falling edges of A nd B meaning any logical change occurs we get the number of the pulses. The counter incrementation or decrementation depends on which channel is leading the other.
- X4 Decoding used by either hardware or software, and here we made it through software (coding).
- We use two external interrupts, both interrupts will occur at any logical change, with every interrupt pulse count will increment or decrement depending on the direction of the encoder (which channel leads the other) and the angle will be measured through this equation :

Amount of Rotation is $(^\circ) = \frac{Edge\_Count}{xN} \cdot 360^\circ$

Where N equals the number of pulses generated or counted by the encoder per shaft revolution
And x = Decoding type

Chapter 6

Motion planning control

1- Introduction:

The fundamentals of motion control focus on the details about motor control, feedback methods and mechanical actuators.

In this project we need to control the motion of the vehicle all the time in order to make the vehicle follow the specific path and trajectory of position, speed and acceleration.

The kinematics model of the vehicle system is very important in order to control the vehicle motion.

2- ATV Steering system:

The ATV steering system is based on Ackermann steering system fig 6.1.

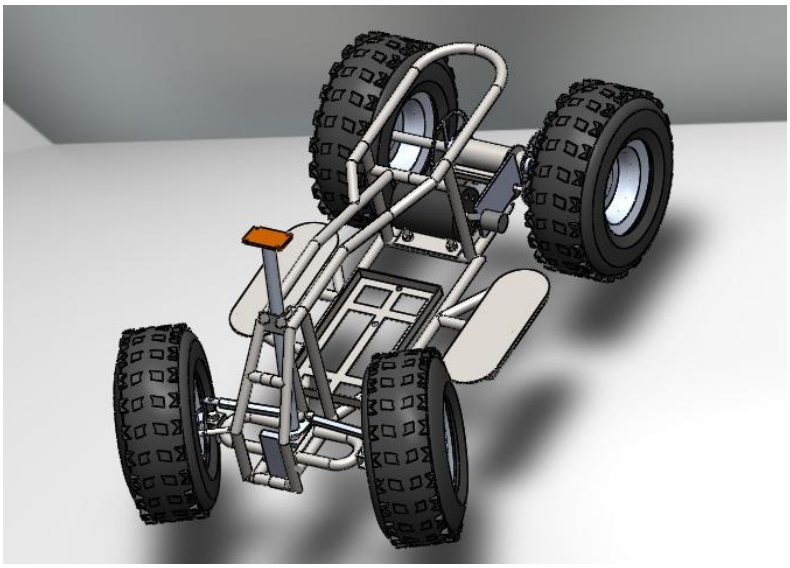


FIGURE6. 1: ATV ACKERMANN STEERING SYSTEM

2.1- Ackermann steering:

The Ackermann principle is based on the two front steered wheels being pivoted at the ends of an axle-beam. The original Ackermann linkage has parallel set track rod arms, so that both steered wheels swivel at equal angles. Consequently, the intersecting projection lines do not meet at one point (Fig. 6.2 If both front wheels are free to follow their own natural paths, they would converge and eventually cross each other. Since the vehicle moves along a single mean path, both wheel tracks conflict continuously with each other causing tire slip and tread scrub. Subsequent modified linkage uses inclined track-rod arms so that the inner wheel swivels about its king pin slightly more than the outer wheel. Hence the lines drawn through the stub-axles converge at a single point somewhere along the rear-axle projection (Fig. 6.3).

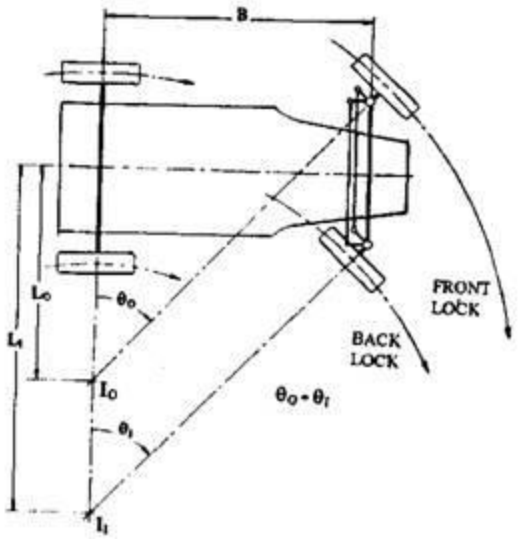


FIGURE6. 2: SIDE-PIVOT STEERING WITH PARALLEL-SET TRACK-ROD ARMS

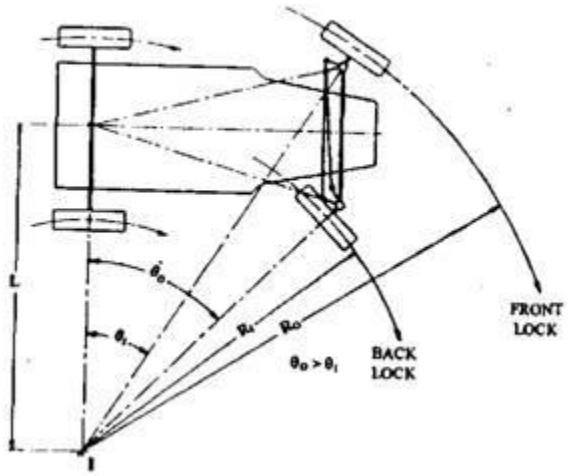


FIGURE6. 3: SIDE-PIVOT STEERING WITH INCLINED TRACK-ROD ARMS.

3- Kinematics model:

First we need to define the kinematics model of the vehicle, the kinematics model describe the motion and position of the vehicle be define the relation between the wheel velocity and (X,Y,θ) pose of the vehicle.

3.1- Forward kinematics model of Ackermann:

An Ackermann steering vehicle can be suitably represented, for the sake of designing the trajectory tracking controller, by a kinematic bicycle model (Figure 1). Considering the center of the rear wheel of the bicycle, its motion is described by the following equations:

$$\dot{X} = v \cdot \cos\theta$$
$$\dot{Y} = v \cdot \sin\theta$$
$$\dot{\theta} = \frac{v}{L} \cdot \tan\delta$$

where (x,y,θ) define the vehicle pose, i.e. the position and orientation of the rear wheel-frame in the Cartesian plane, v the vehicle linear velocity , δ is the steering angle and L is vehicle wheel base.

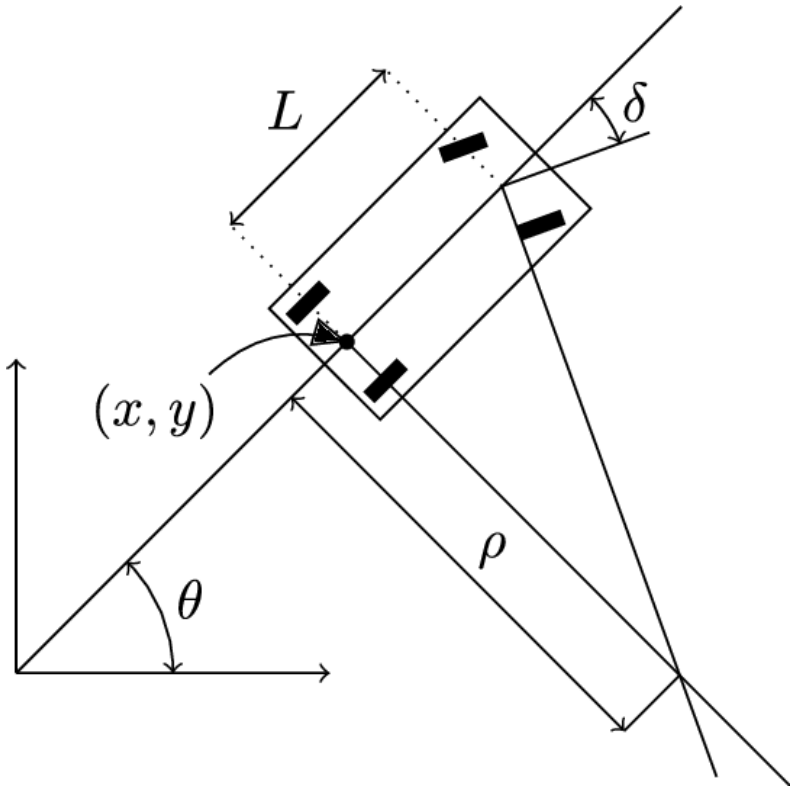


FIGURE6. 4: KINEMATIC BICYCLE MODEL

With forward kinematics we can find the position and the orientation of the vehicle if we know the linear velocity and the steering angle.
To find the pose (position and orientation) of the vehicle we need to integrate the previous equations, so the full mathematical model of the vehicle shown in these equations.

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f} * \delta_t * dt$$

$$v_{t+1} = v_t + a_t * dt$$

Now we can find the pose of the vehicle (X, Y, ψ) if we know the (a) acceleration and (δ) steer angle of the vehicle and of course (dt) the sampling time of the system.

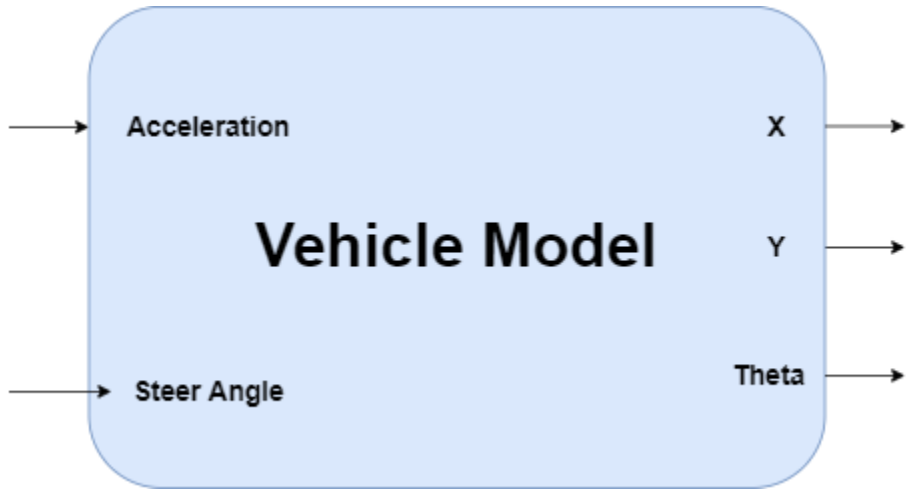


FIGURE6. 5: MATH VEHICLE MODEL

4- Path model:

In order to make the vehicle follow a path or a trajectory of positions, we need to mathematically model the path.

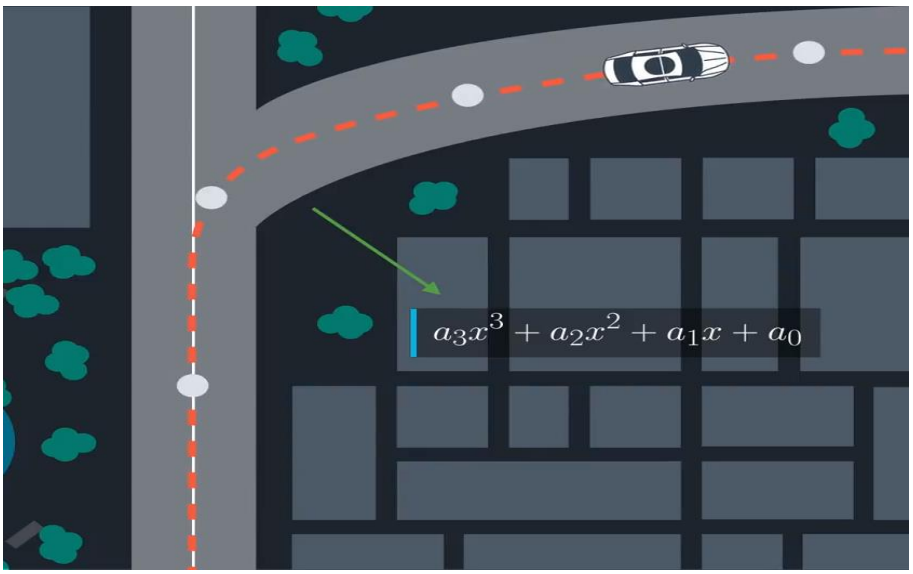


FIGURE6. 6: PATH MODELING

We will model the path a third order polynomial, this polynomial can easily fit to the real word road and highways.
To find the best orientation (ψ) of the vehicle at each (X, Y) position in the path we need to find the slope or the gradient of the path by differentiate the polynomial and take the arctan of the derivative.

$$y = f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$tan(\psi_{dest}) = \frac{dy}{dx} = 3a_3x^2 + 2a_2x + a_1x$$

5- Pose Errors:

With the vehicle model and the path model we can calculate and define the error in the vehicle position, there are two type of error, first the cross-track error (CTE): the perpendicular distance between the CG of the vehicle and the path.

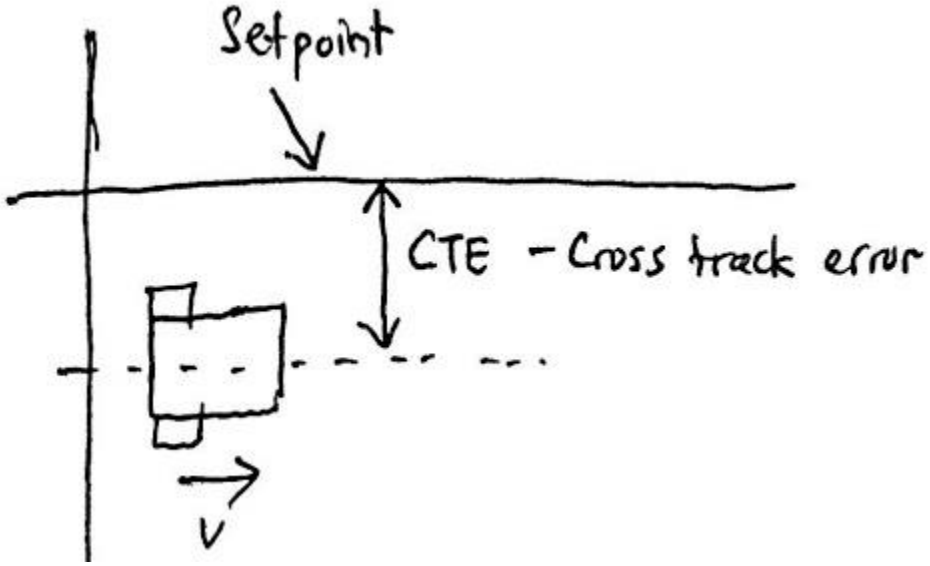


FIGURE6. 7: THE CROSS-TRACK ERROR (CTE)

The second type of error is the heading error: the difference between the heading of the vehicle (orientation) and the reference angle from the path arctan derivative.

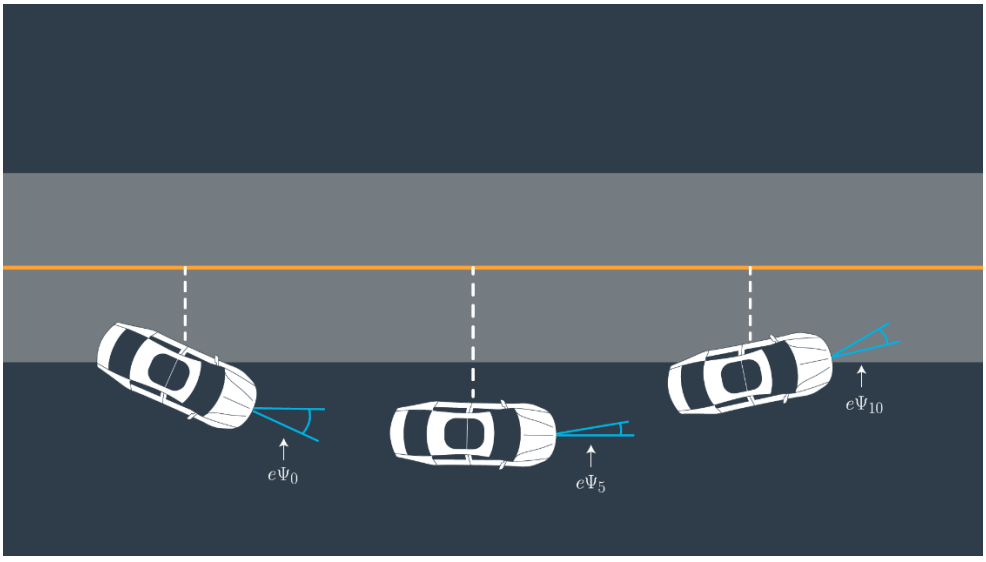


FIGURE6. 8: THE HEADING ERROR

The two errors can be calculated with these two equations:

$$cte_{t+1} = f(x_t) - y_t$$

$$e\psi_{t+1} = \psi_t - \psi des_t$$

While the f(x) is the desired Y position of the vehicle and (ψ-des) is the desired heading of the vehicle.

6- Controller design:

Now we have a full mathematical definition of the system make us able to design a controller that make the vehicle follow the path with minimum errors.

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f} * \delta_t * dt$$

$$v_{t+1} = v_t + a_t * dt$$

$$cte_{t+1} = f(x_t) - y_t + (v_t * sin(e\psi_t) * dt)$$

$$e\psi_{t+1} = \psi_t - \psi des_t + (\frac{v_t}{L_f} * \delta_t * dt)$$

But first we need to notice some characteristic of the vehicle system that will be important in designing and selecting the best controller, first the vehicle system is multi input multi output (MIMO), second the vehicle system has constrains in the inputs, steer angle have range of (-30, 30) degree and acceleration range (-1, 1).

7- Model predictive controller:

MPC uses the mathematical model of the vehicle system to make predictions about the vehicle future behavior or position. MPC solves an online optimization algorithm to find the optimal control action (steer angle, acceleration) that drives the predicted output to the reference trajectory. MPC can handle multi-input multi-output systems that may have interactions between their inputs and outputs. It can also handle input and output constraints which make it very suitable for controlling the vehicle. MPC has preview capability; it can incorporate future reference information into the control problem to improve controller performance, which means it not only deal with the present error in deal with future errors that he predicts.

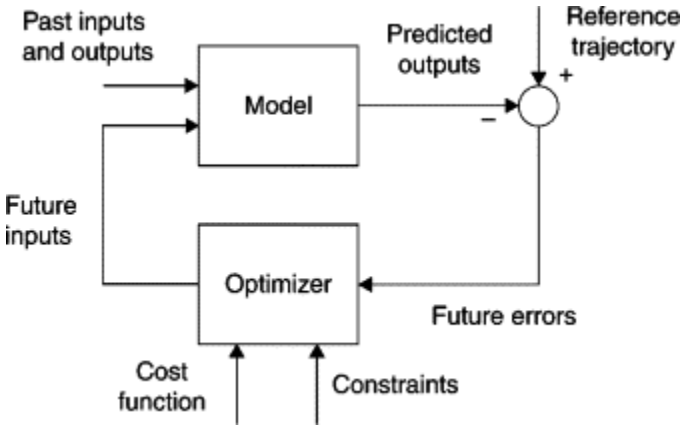


FIGURE6. 9: MPC OPERATION

The only downside of the MPC is the computation cost of the online optimization algorithm, so it needs a powerful computer to work in real-time performance, and also need to be written in high performance programming language like C++ / C.

The MPC optimizer need a cost function that define the total error of vehicle state (position, velocity) to minimize it, the optimizer finds the best set of in N inputs (steer angle, acceleration) that minimize the cost function.

In MPC, we define a cost function to optimize our path with the trajectory. For speed, we penalize the model if the car cannot maintain at a target speed. We want no acceleration and zero steering if possible. But since it is unavoidable, we want the rate change to be as low as possible if it happens. This reduces motion sickness and saves gas. In our model, our cost includes:

- Cross-track error
- Heading error
- Speed cost (prefer staying at say 1 m/s)
- Steering cost (prefer zero steering)
- Acceleration cost (prefer zero acceleration)

Since those objectives may be contradictory to others, we add weights to the cost to reflect its priority. Here is the cost function:

$$J = \sum_{t=1}^{N} w_{cte}\|cte_t\|^2 + w_{e\psi}\|e\psi_t\|^2 + w_v\|v_t - v_{target}\|^2 + \sum_{t=1}^{N-1} w_\delta\|\delta_t\|^2 + w_a\|a_t\|^2 + \sum_{t=2}^{N} w_{rate_\delta}\|\delta_t - \delta_{t-1}\|^2 + w_{rate_a}\|a_t - a_{t-1}\|^2$$

Now the MPC will find the best N inputs to the vehicle, the N is the prediction horizon, it show how far the MPC can predict in the future and (dt) is the sampling time between each input signal, so the horizon time:
T = N * dt

Note: the larger N the more computation cost.
Note: the smaller dt the more computation cost.

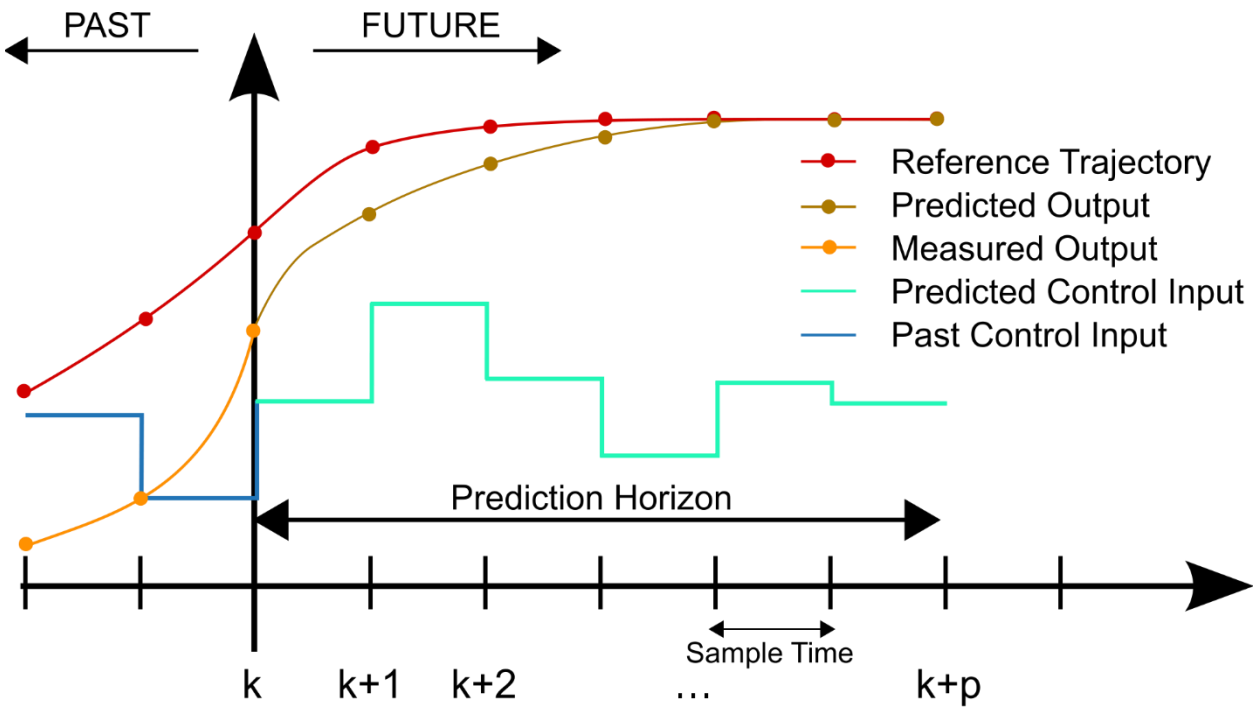


FIGURE6. 10: MPC PREDICTION HORIZON

8- Control block diagram:

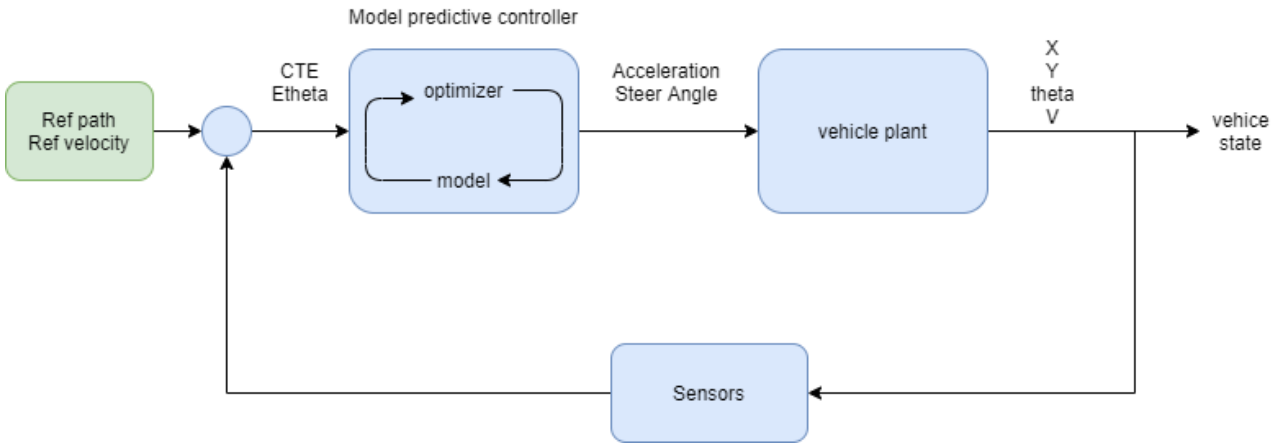


FIGURE6. 11: MPC CONTROL BLOCK DIAGRAM

MPC algorithm run in these steps:

1- Define N, dt, T.
2- Define vehicle model and constrains.
3- Define the cost function.
4- Read the velocity, heading angle and X,Y position from sensors and pass them to the MPC.
5- Pass the current state of the vehicle to MPC optimizer.
6- The MPC optimizer find the best N actuation signal (steer angle, acceleration).
7- Apply only the first control inputs.
8- Repeat from step 4.

9- Results:

Note: the error of the position is proportional to the reference velocity of the vehicle and the angular velocity of the steer angle.

The image below is a real-time plot of the vehicle position.

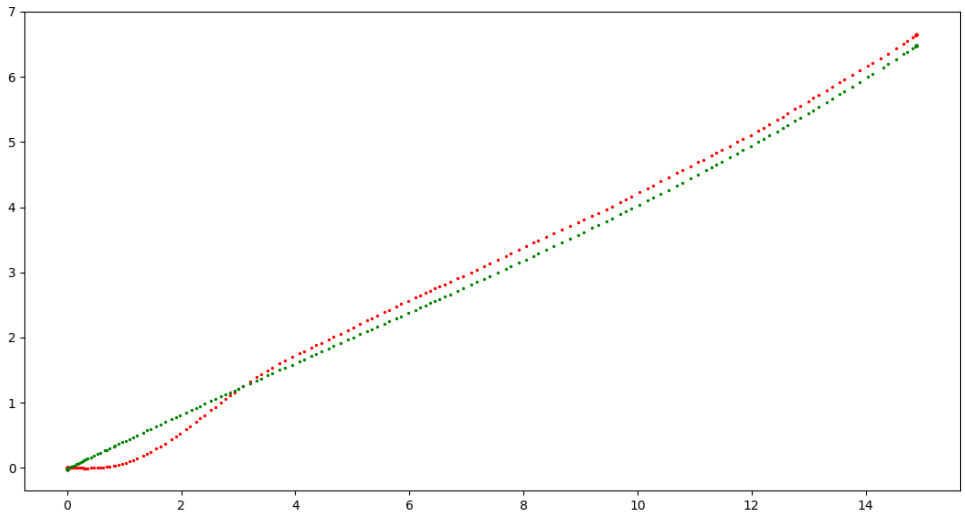
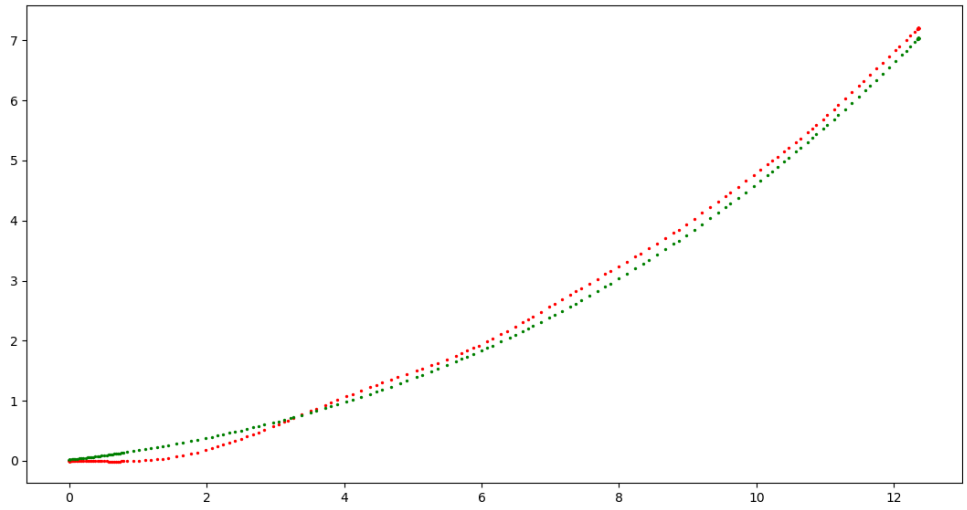
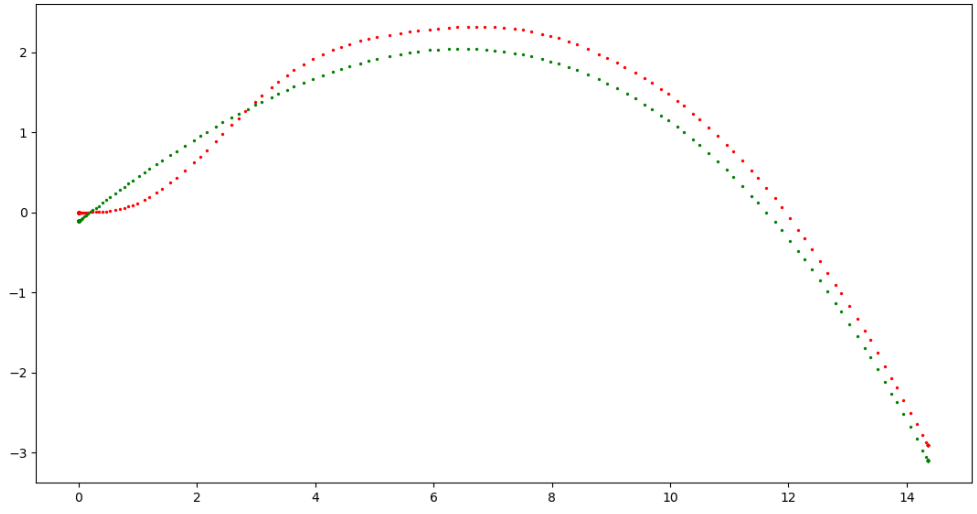


FIGURE6. 12: POSITION REAL-TIME PLOT


0
1
2
3
4
5
6
7
0
2
4
6
8
10
12


2
1
0
−1
−2
−3
0
2
4
6
8
10
12
14

Chapter 7

Localization

Introduction for localization

To make an accurate localization we should use more than one sensor. In our project we used three sensors they are GPS, IMU and Encoder to collect data from different sources as there isn't a 100% accurate sensor, we can know the accurate position of the vehicle by integrating data together. From encoder we get the displacement the vehicle moved and velocity. From IMU we get the relative position, acceleration, and the heading of the car. From GPS we get the absolute position.

1- MPU 9250

This MPU is a 9-axis motion tracking device that have a 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, and a DMP (Digital Motion Processor). The MPU-9250 also have an embedded temperature sensor.
This module features the MPU-9250, which is a multi-chip module (MCM) consisting of two dies integrated into a single quad-flat no-leads (QFN) package. First die have the 3-Axis accelerometer and the 3-Axis gyroscope. The other die have the 3-Axis magnetometer AK8963. The polarity of the two dies shown in figures below. Note: pin-1 identifier (•) in the figure.

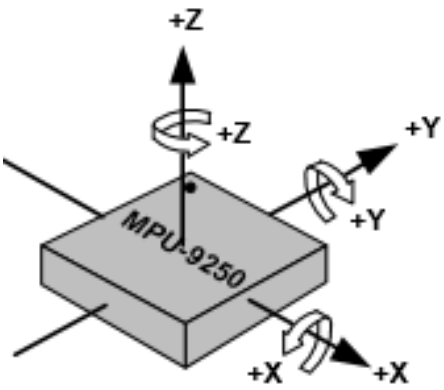


FIGURE7. 1: MPU ORIENTATION OF AXES OF SENSITIVITY AND POLARITY OF ROTATION FOR ACCELEROMETER AND GYROSCOPE.

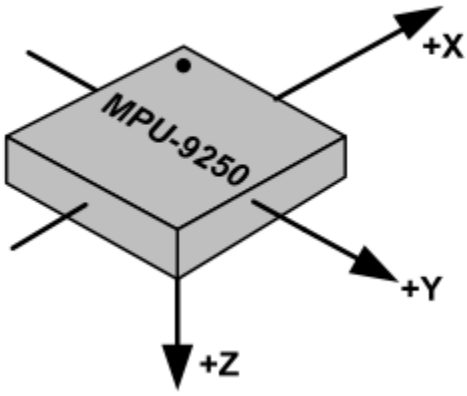


FIGURE7. 2: ORIENTATION OF AXES OF SENSITIVITY FOR COMPASS

1.1- MPU 9250 Specifications

* On-board pull-up resistors on SDA, SCL, and nCS
* On-board pull-down resistors on FSYNC and AD0
* 3-Axis Accelerometer
* Range: up to ± 16 g
* Sensitivity: up to 16,384 LSB/g
* 3-Axis Gyroscope
* Range: up to ± 2000 deg/sec
* Sensitivity: up to 131 LSB/deg/sec
* 3-Axis Magnetometer
* Range: ± 4800 μT
* Sensitivity: 0.6 μT/LSB
* Supply Voltage: 4.4 to 6.5 V or 3.3V if you solder the solder jumper near the on-board voltage regulator
* Interface: I2C
* 2C Address: 0x68 by default, 0x69 if AD0 is pulled high
* Board Dimensions: 25.5mm (1.004") long x 15.4mm (0.606") wide, 3mm (0.118") inside diameter of mounting holes
* Weight: 2.72g (0.096oz)

1.2- MPU and Arduino hardware connection

MPU Module	Arduino
SDA	A4
SCL	A5
VCC	3.3V
GND	GND

As shown in figures below.

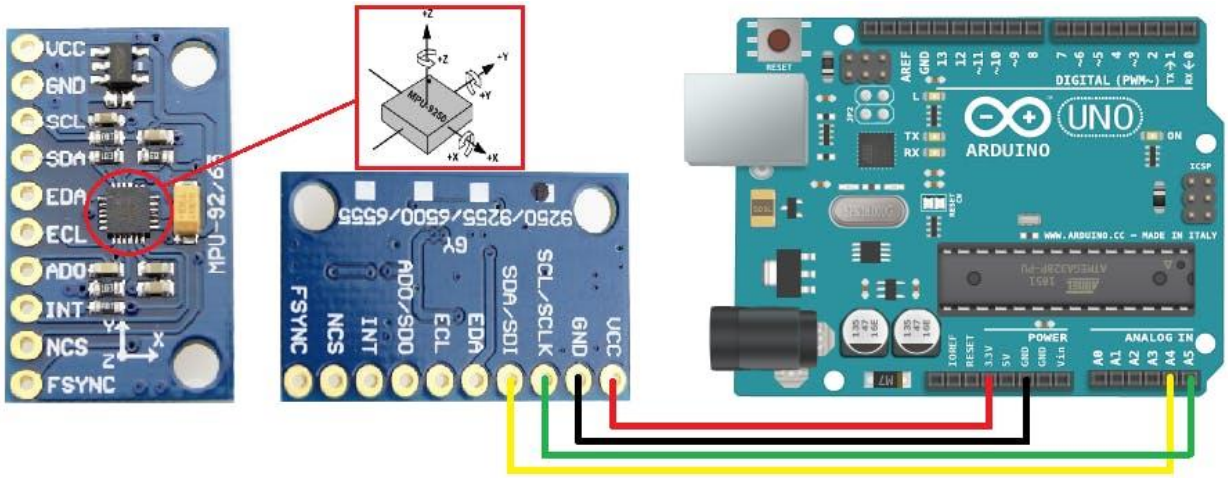


FIGURE7. 3: ARDUINO AND MPU CONNECTION

As shown in figure below A4 & A5 are pins for I2C.

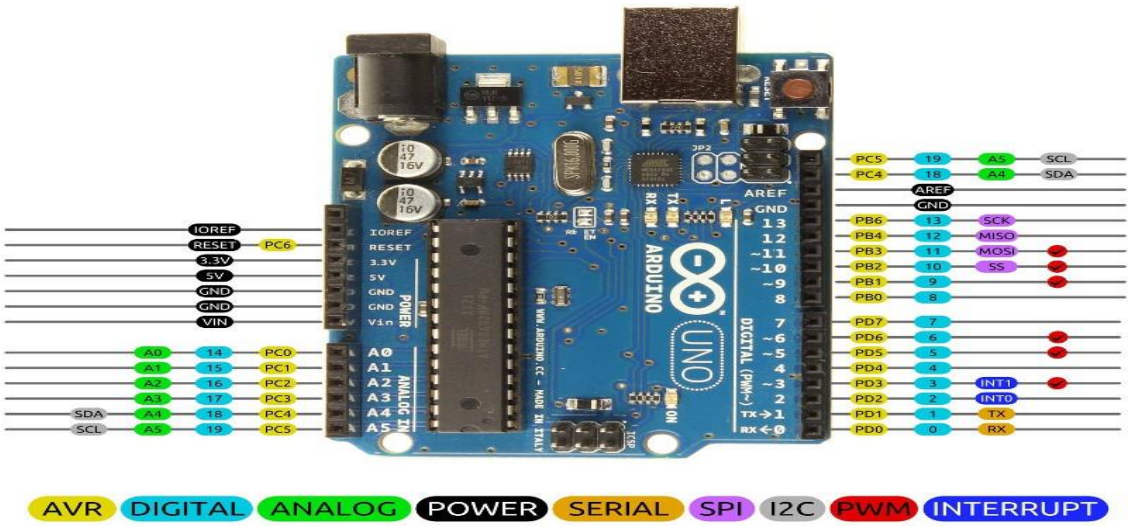


FIGURE7. 4: ARDUINO UNO PINOUT

1.3- Code explanation

First we make calibration then get the data from the sensor. We are only interest in yaw angle and magnetometer.

For yaw angle and we have accelerometers which respond slowly but are accurate over time, we have Gyros which respond quickly, but drift over time. We merge these two sensor readings to give us a quick response which is also accurate.

All of this was theoretical, but when it came to practical there was a little drift even after merge, this drift over time ruin the reading completely and if there is any vibration from moving it makes it worst.

We solve this problem by adding a filter, its idea was to pass only values greater than the noise factor we choose. As shown in figure below.

```
if (angle-Langle > 0 && angle - Langle < noise_factor )
{
  angle = Langle;
}

else if (angle-Langle < 0 && angle - Langle > -noise_factor )
{
  angle = Langle;
}

Langle = angle;
```

FIGURE7. 5: MPU DRIFT SOLUTION FILTER

As the sensor drift and the vibration action was less than one degree, so the noise factor we choose was small and doesn't affect the value of the angel we need to measure, as the needed accuracy is one degree. And here is a graph to show the output at zero position before and after the filter. As shown in figures below.

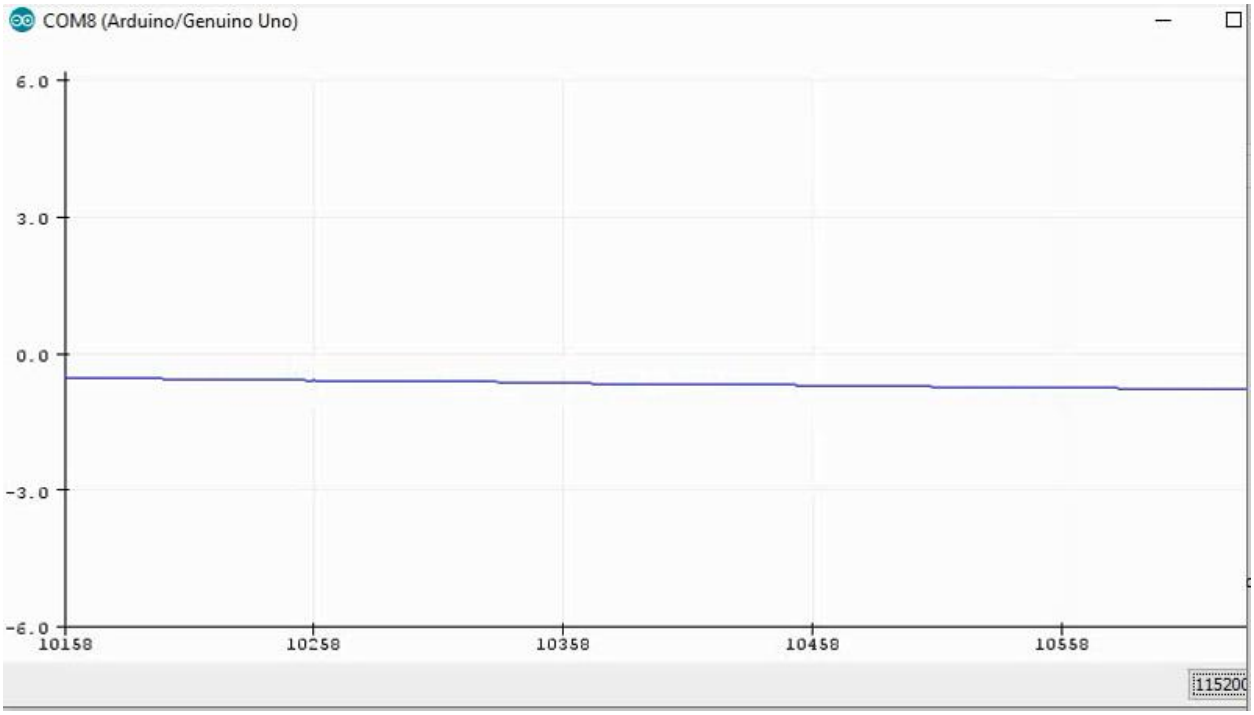


FIGURE7. 6: MPU YAW ANGLE AT ZERO POSITION BEFORE FILTER

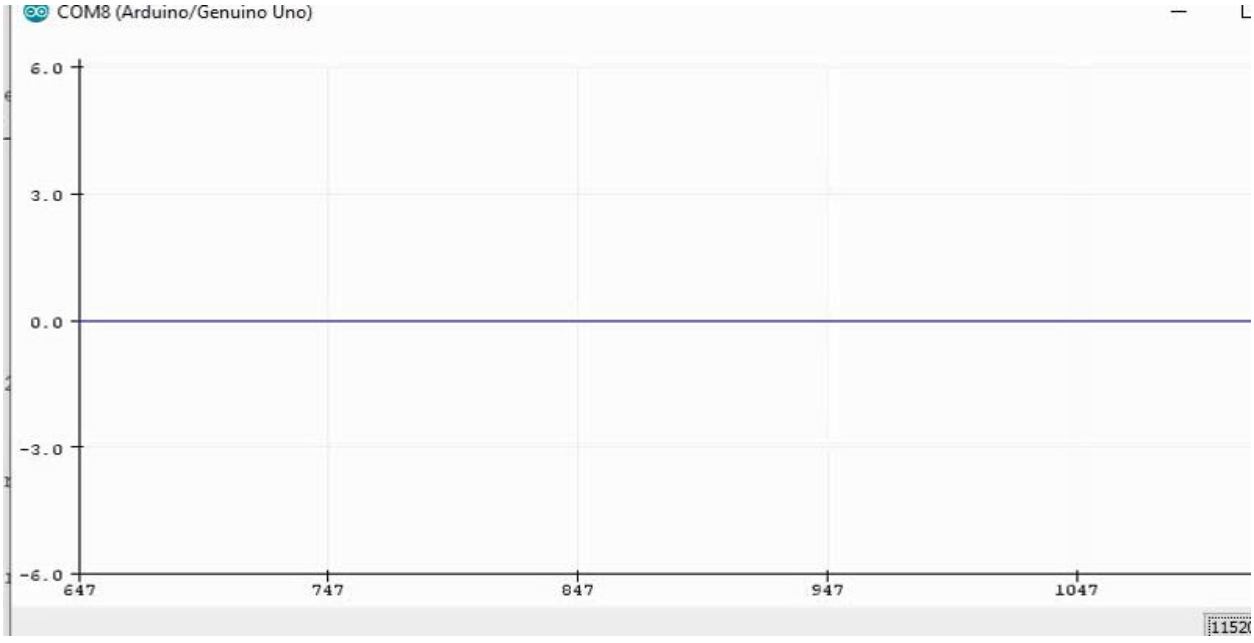


FIGURE7. 7: MPU YAW ANGLE AT ZERO POSITION AFTER FILTER

For magnetometer we read the magnetic field in x-direction and y- direction. Then we use this equation to get magnetic field in horizontal direction.

MagHorizDirection= atan2 (magX, magY) * $\frac{180}{\pi}$

2- NEO-6M GPS

The NEO-6M GPS module is a well-performing complete GPS receiver with a built-in 25 x 25 x 4mm ceramic antenna, which provides a strong satellite search capability. With the power and signal indicators, you can monitor the status of the module. Thanks to the data backup battery, the module can save the data when the main power is shut down accidentally. As shown in figure below.

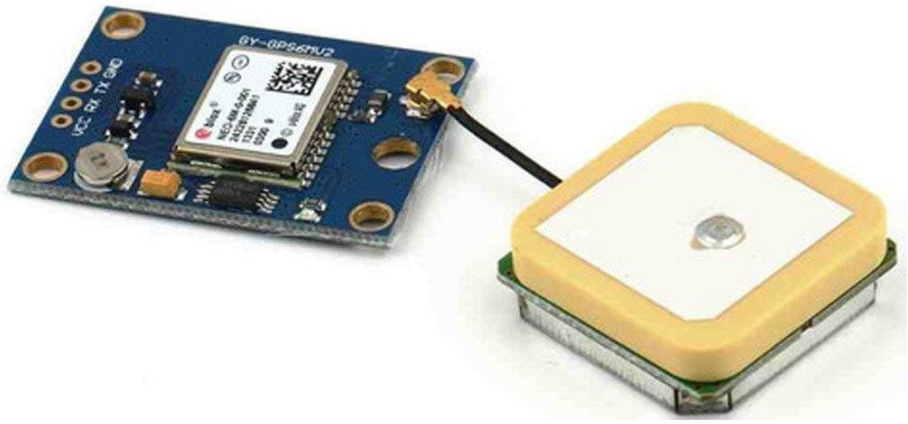


FIGURE7. 8: NEO-6M GPS

2.1- GPS specifications

- Cold start time of 38 s and Hot start time of 1 s
- Supply voltage: 3.3 V t
- Configurable from 4800 Baud to 115200 Baud rates. (default 9600)
- Super Sense Indoor GPS: -162 dBm tracking sensitivity
- 5Hz position update rate
- Operating temperature range: -40 TO 85°C
- UART TTL socket
- EEprom to save configuration settings
- Rechargeable battery for Backup
- Separated 18X18mm GPS antenna
- **Dimension:** 22X30X13 mm
- Weight: 12g

2.2- GPS Features

- A complete GPS module with an active antenna integrated, and a built-in EEPROM to save configuration parameter data.
- Built-in 25 x 25 x 4mm ceramic active antenna provides strong satellite search capability.
- Equipped with power and signal indicator lights and data backup battery.
  Power supply: 3-5V; Default baud rate: 9600bps.
- Interface: RS232 TTL

2.3- Hard ware connection

Connect the NEO-6M GPS Module and the TTL Module as shown below

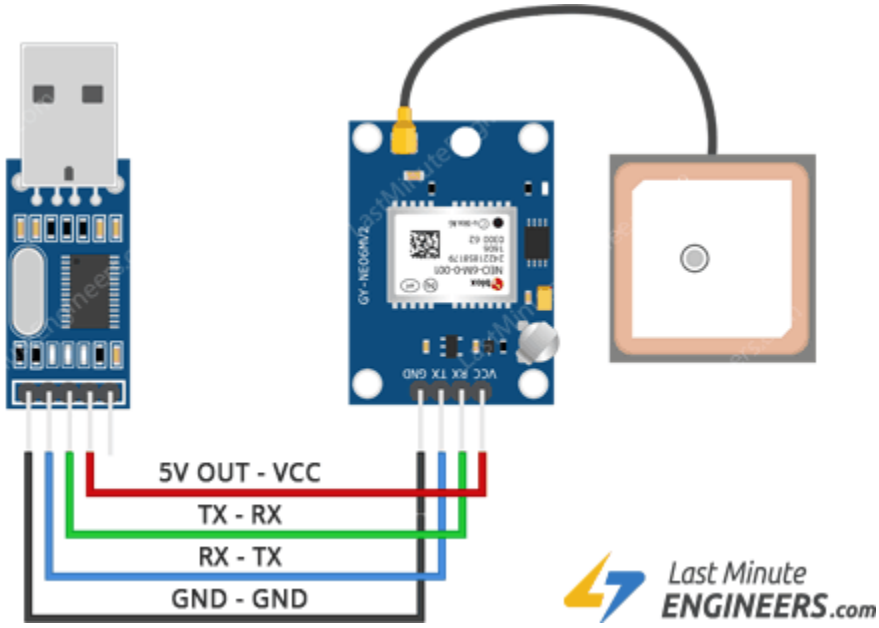


FIGURE7. 9: GPS AND TTL CONNECTION

NEO-6M GPS Module	TTL Module
TX	RX
RX	TX
VCC	VCC
GND	GND

2.4- Code explanation

- We use TTL for communication with GPS directly through UART protocol which is Full duplex protocol that's mean it can sent and receive at same time, and also its Asynchronous protocol which means it doesn't need clock , agree on speed of transmission and receiving before transfer data.
- The TTL in connected directly to the computer, and we use an application called u-block to see the GPS output and its accuracy.
- As shown in figure below, this application shows the accuracy of the GPS output for the same position during a period of time in 10 meter area, collecting data every one second, if the accuracy of the GPS was 100% it should give the same point every time but as shown there is a deviation and the values are varying but mainly repeated in 6 meters. So we can say its accuracy about 6 meters.

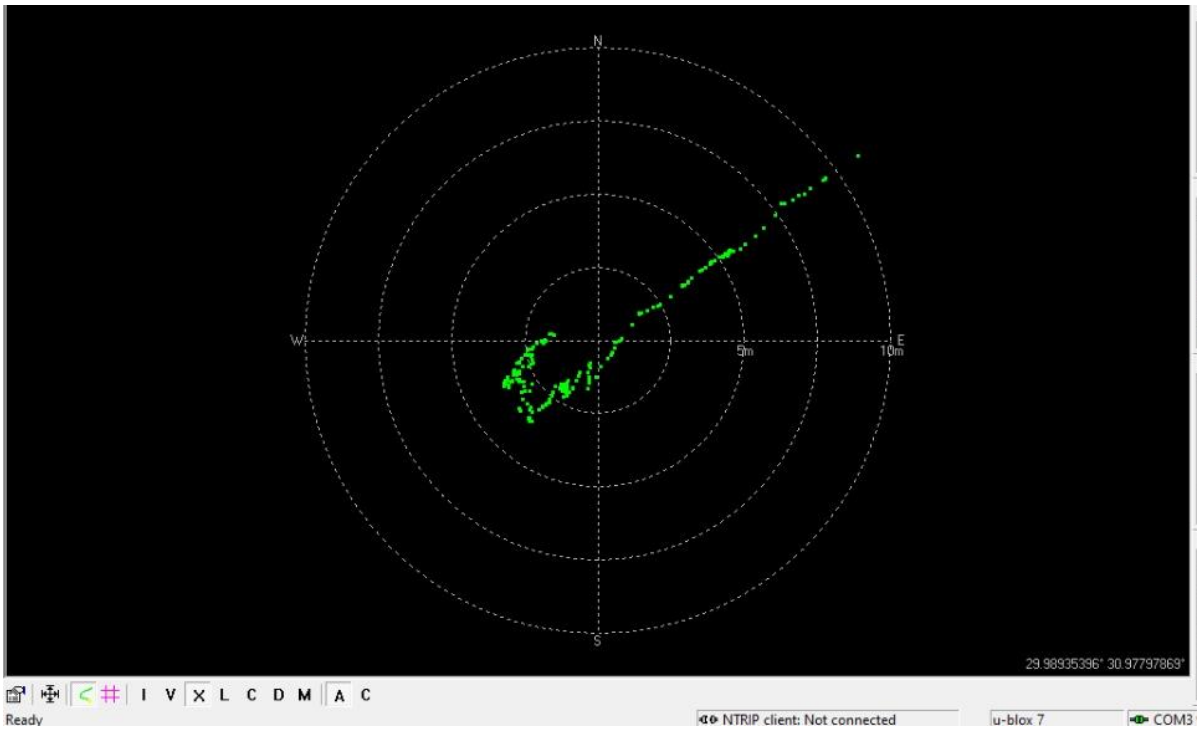


FIGURE7. 10: U-BLOX GPS OUTPUT

The figure below shows the GPS output is longitude and latitude and how many satellites are connected to our GPS & their nationalities.

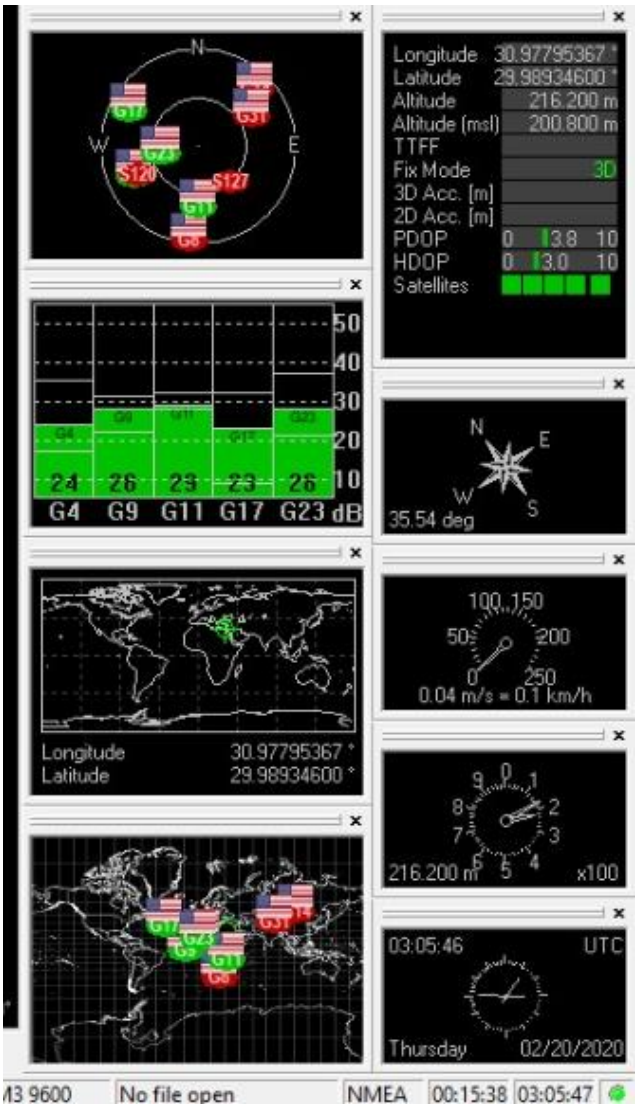


FIGURE7. 11: U-BLOX GPS OUTPUT

In our project we need to know a distance between two points in metric, we have already done this using an encoder, but encoder gives a value relative to the starting point.
Therefore we used GPS to get the absolute values, the GPS outputs longitude and latitude aren't in meters so we can't directly get the distance from this numbers. We convert longitude and latitude to **UTM** (Universal Transverse Mercator), so we can get distance in (x, y) coordinate by using ROS package.

2.5- UTM (Universal Transverse Mercator)

A UTM zone is a 6° division of the Earth. As circle has 360°, this means that there are 60 UTM zones on Earth. (360 ÷ 6 = 60).

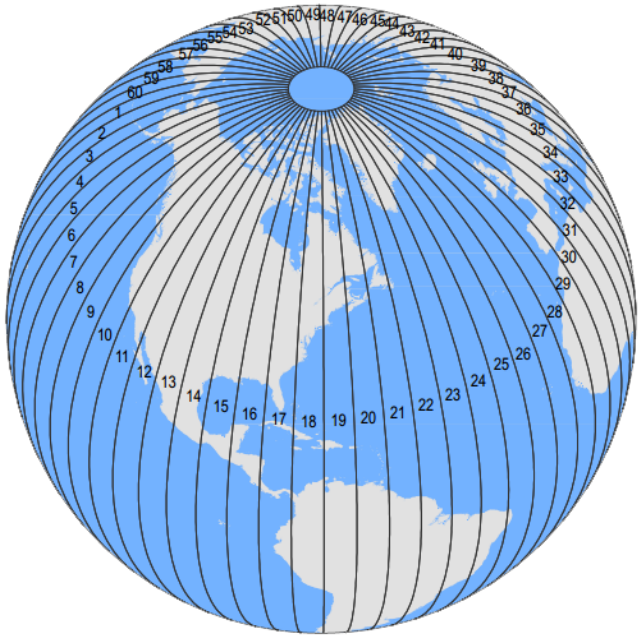


FIGURE7. 12: UTM ZONES

As a replacement for using **latitude and longitude coordinates**, each 6° wide UTM zone has a **central meridian of 500,000 meters**. This central meridian is an arbitrary value suitable for avoiding any negative easting coordinates. All easting values east and west of the central meridian will be positive. As shown in figures below

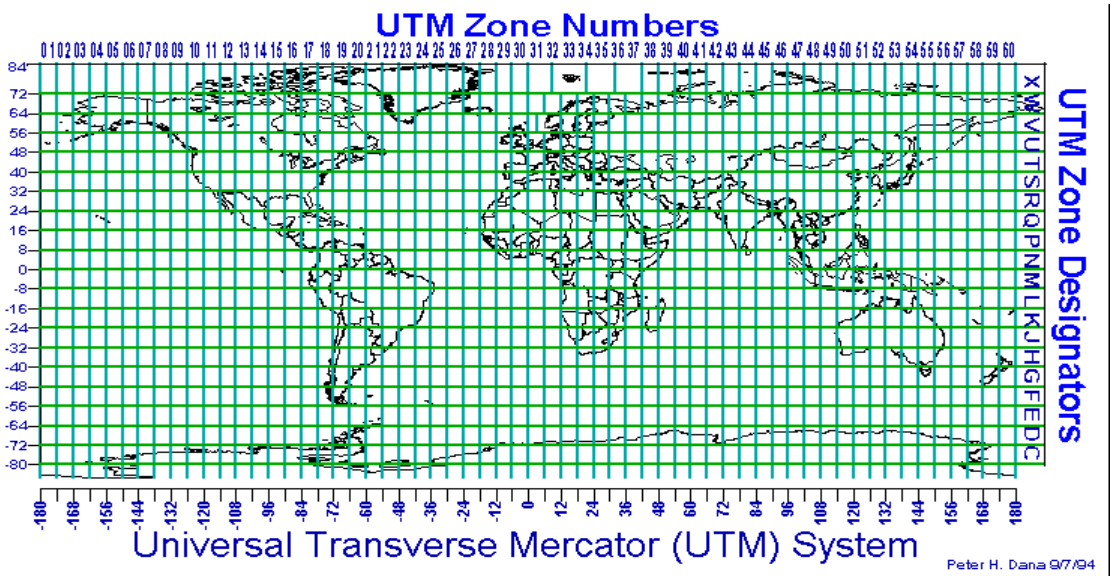


FIGURE7. 13: UTM ZONE NUMBERS

If we are in the northern hemisphere, the equator has a northing value of 0 meters.

In the southern hemisphere, the equator starts at 10,000,000 meters. To make all values south of the equator to be positive. This is named a false northing because y-coordinates in the southern geographic area will avoid negative values.

In our country Egypt our zoon is in Northeast so out coordinates measured in y-axis from zero in equator, and x-axis from 500,000. As shown in figure below.

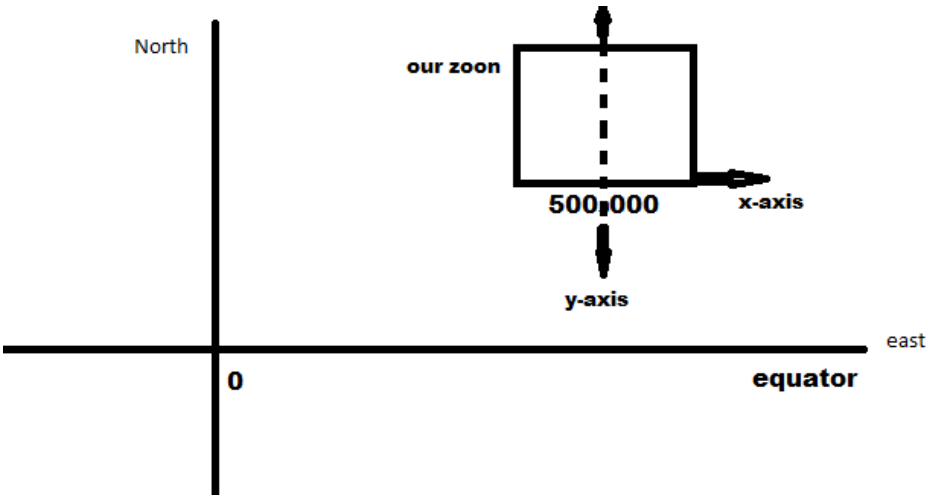


FIGURE7. 14: EGYPT ZONE AXIS IN UTM SYSTEM

3- Odometry

- The idea of Odometry is the use of data from motion sensors to estimate change in position over time.

We use odometry to get (x ,y) coordinate using the velocity we get from encoder and the angle we get from MPU.
Using this two equations:

$$x_t = x_{(t-1)} + v cos\theta * dt$$
$$y_t = y_{(t-1)} + v sin\theta * dt$$

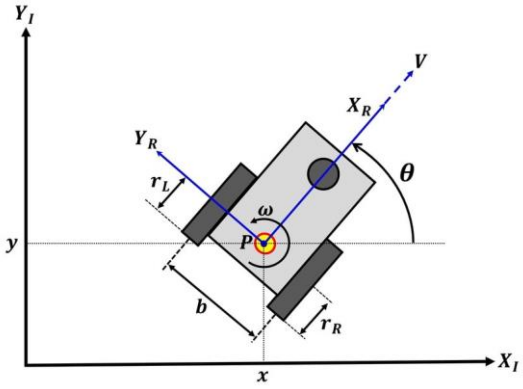


FIGURE7. 15: ODOMETRY

Where
V: is linear velocity
θ: Gyro angle
dt: sampling time

$x_{(t-1)}$,$y_{(t-1)}$: is the previse distance in x&y

4- Localization by sensor fusion

4.1- Why we use imu with gps by sensor fusion not gps only?

The gps will be get a few meters off but gps might have to paired with additional imu sensor to get the accuracy to we need to give you a more visual sensor.

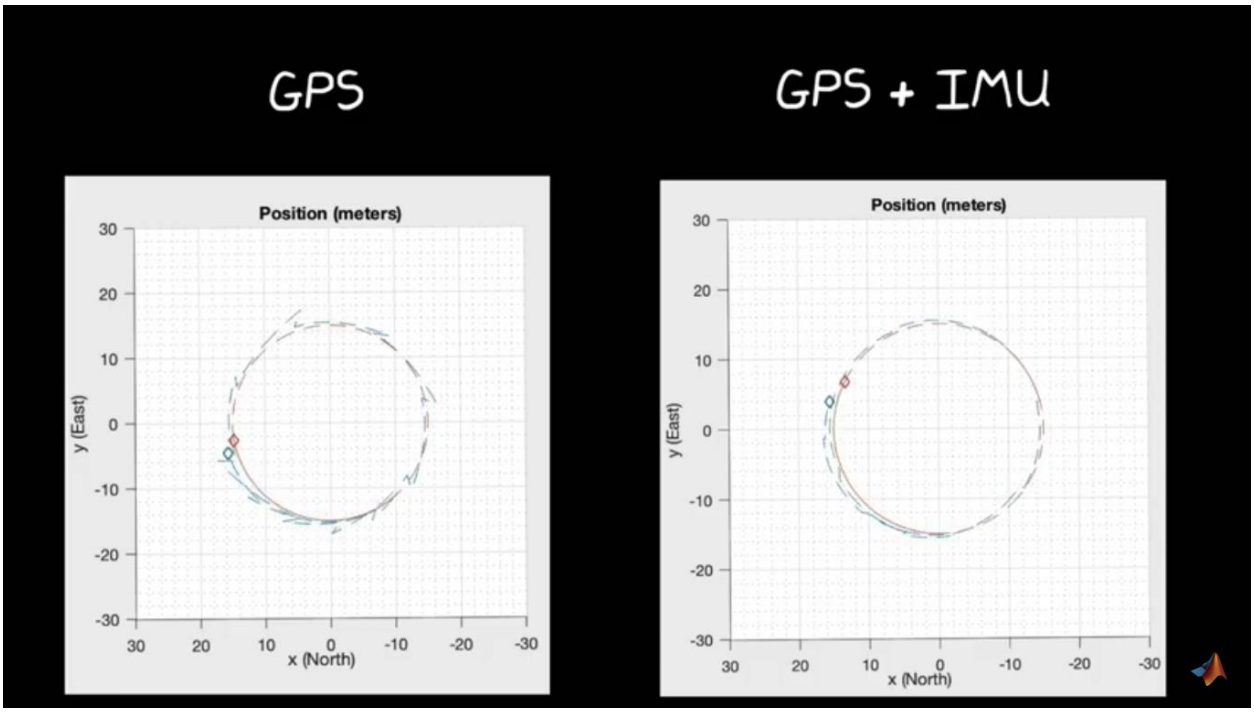


FIGURE7. 16: GPS AND IMU COMPARSION

In the figure shown, a comparison between the GPS used only and the GPS with imu in the red path is the true trajectory and the blue trajectory is the trajectory from the sensor:

- **In GPS only,** we find that the trajectory with the sensor deviates from its true trajectory, because the GPS will get a few meters as we explained earlier.

- **When adding imu to the GPS,** we find that the trajectory with the sensors achieves the true trajectory, because by adding imu to the GPS we will improve the accuracy.

4.2- GPS and IMU complement each other by:

GPS	IMU
Low frequency	High frequency
Low accuracy	High accuracy
Absolute position	Drift
Used to update position	Used to predicated position

4.3- How to achieve sensor fusion between imu and GPS?

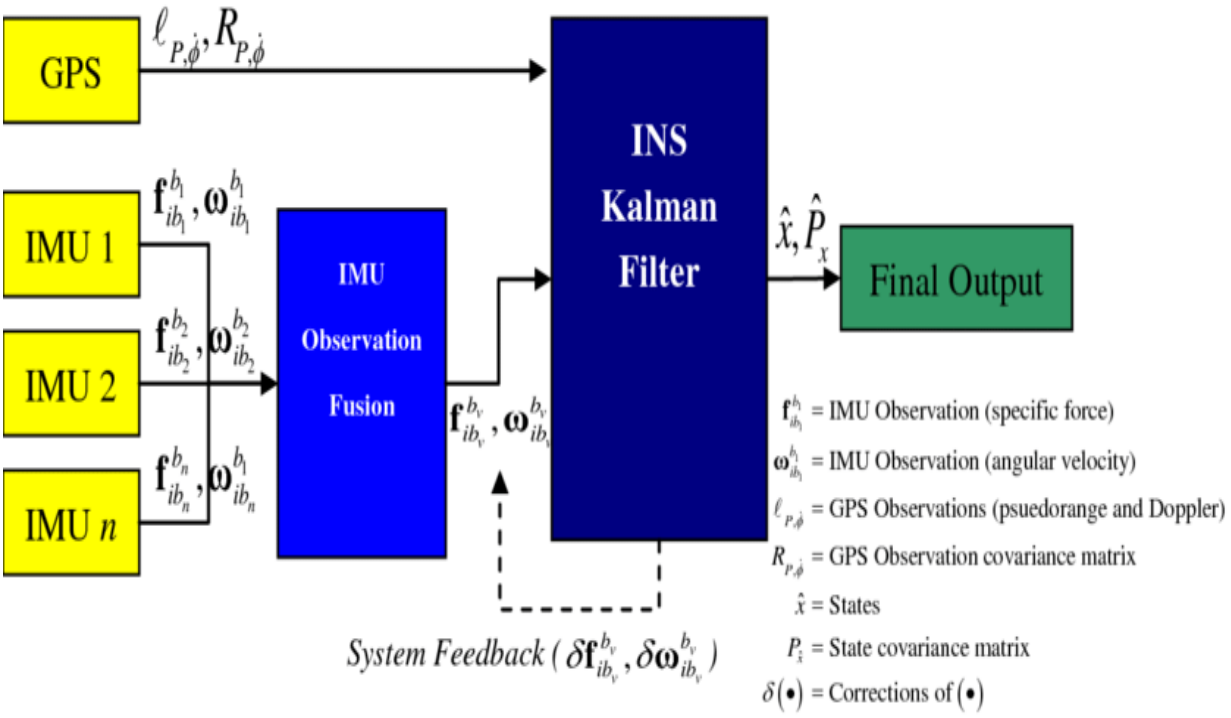


FIGURE7. 17: KALMAN FILTER PROCESS

In the figure shows the steps to implement sensor fusion:

1) Find the average between the group of imu.
2) We do sensor fusion between gps and imu by Kalman filter.
3) We get final output from kalman filter as state position and state covariance matrix.

5- Kalman filter

5.1- What is a kalman filter?

In statistics and control theory, Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

5.2- Why did we use a Kalman filter?

Because It is an iterative mathematical process to quickly estimate the true value, position, velocity.

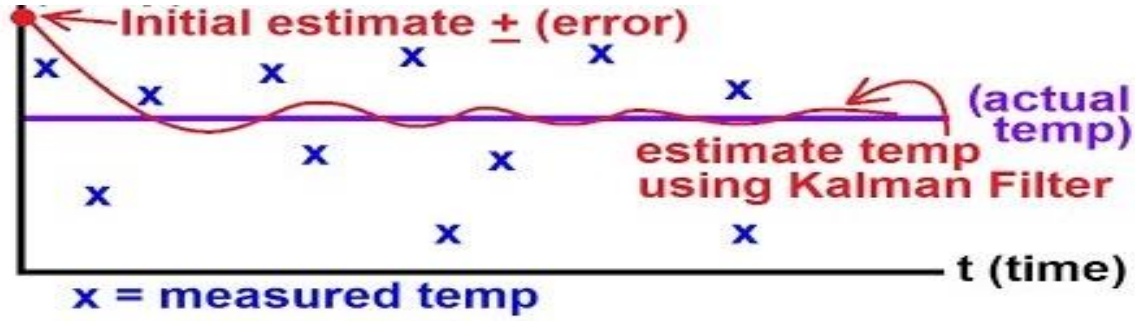


FIGURE7. 18:KALMAN FILTER ESTIMATION

5.3- How Kalman filter work?

Kalman filter algorithm consists of two stages: prediction and update.
Note that the terms “prediction” and “update” are often called “propagation” and “correction,” respectively, in different literature. The Kalman filter algorithm is summarized as follows:

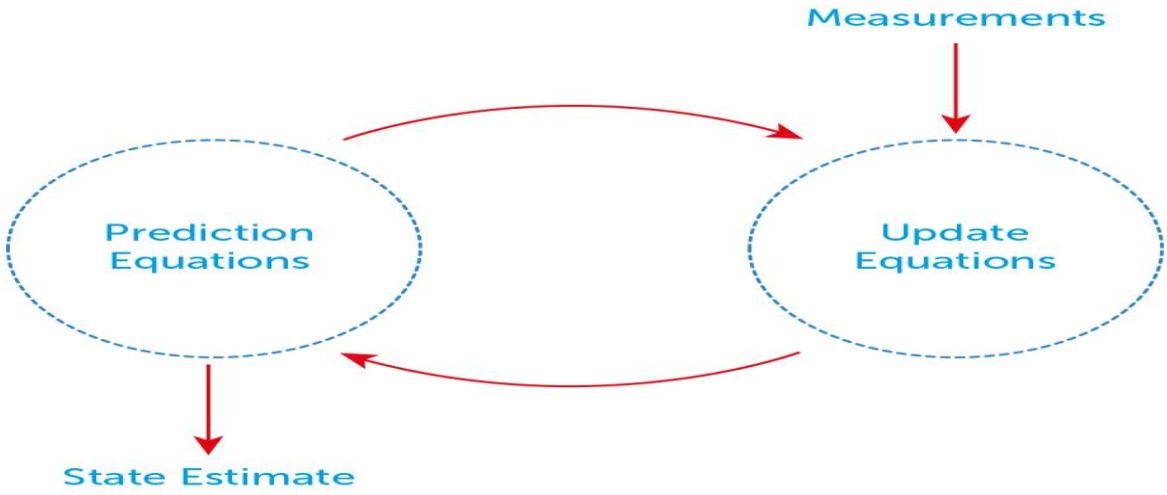


FIGURE7. 19: KALMAN FILTER ALGORITHM

5.4- Extended kalman filter

The extended Kalman filter provides us a tool for dealing with such nonlinear models in an efficient way. Since it is computationally cheaper than other nonlinear filtering methods such as point-mass filters and particle filters.
The extended Kalman filter can be viewed as a nonlinear version of the Kalman filter that linearized the models about a current estimate.

The Equations used in extend Kalman filter

BLUE = inputs ORANGE = outputs BLACK = constants GRAY = intermediary variables

State Prediction
(Predict where we're gonna be)

$$\mathbf{x}_{predicted} = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}\mathbf{u}_n$$

Covariance Prediction
(Predict how much error)

$$\mathbf{P}_{predicted} = \mathbf{A}\mathbf{P}_{n-1}\mathbf{A}^{\mathbf{T}} + \mathbf{Q}$$

Innovation
(Compare reality against prediction)

$$\tilde{\mathbf{y}} = \mathbf{z}_n - \mathbf{H}\mathbf{x}_{predicted}$$

Innovation Covariance
(Compare real error against prediction)

$$\mathbf{S} = \mathbf{H}\mathbf{P}_{predicted}\mathbf{H}^{\mathbf{T}} + \mathbf{R}$$

Kalman Gain
(Moderate the prediction)

$$\mathbf{K} = \mathbf{P}_{predicted}\mathbf{H}^{\mathbf{T}}\mathbf{S}^{-1}$$

State Update
(New estimate of where we are)

$$\mathbf{x}_n = \mathbf{x}_{predicted} + \mathbf{K}\tilde{\mathbf{y}}$$

Covariance Update
(New estimate of error)

$$\mathbf{P}_n = (I - \mathbf{K}\mathbf{H})\mathbf{P}_{predicted}$$

5.5- Parameters for EKF

- Inputs:
  - Un = Control vector. This indicates the magnitude of any control system's or user's control on the situation.
  - Zn = Measurement vector. This contains the real-world measurement we received in this time step.

- Outputs:
  - Xn = Newest estimate of the current "true" state.
  - Pn = Newest estimate of the average error for each part of the state.

- Constants:
  - A = State transition matrix. Basically, multiply state by this and add control factors, and you get a prediction of the state for the next time step.
  - B = Control matrix. This is used to define linear equations for any control factors.
  - H = Observation matrix. Multiply a state vector by H to translate it to a measurement vector.
  - Q = Estimated process error covariance. Finding precise values for Q and R are beyond the scope of this guide.
  - R = Estimated measurement error covariance. Finding precise values for Q and R are beyond the scope of this guide.

5.6- Filter design

- In this simulation, the robot has a state vector includes 4 states at time t.
- xt=[xt,yt,φt,vt] Where x, y are a 2D x-y position, φ is orientation, and v is velocity.
- In the code, "xEst" means the state vector.
- And, PEst is covariace matrix of the state.
- Q is covariance matrix of process noise.
- R is covariance matrix of observation noise at time t.
- The robot has an odometer sensor for speed and a gyro sensor for yaw.

5.7- How calculated parameters

- So, the input vector can be used as each time step: ut=[vt,ωt]
- Also, the robot has a GPS sensor, it means that the robot can observe x-y position at each time: zt=[xt,yt]
- The input and observation vector includes sensor noise.

In the code, "observation" function generates the input and observation vector with noise. (For calculated with in the real we can remove this noise).

5.8- Motion Model

- when the robot are moving we must calcuted in this motion :-
- The robot model is ẋ=vcos(φ) ẏ=vsin((φ)
- When φ=ω
- so the motion model is
- x(t+1) = F*xt + B*ut

Where F and B ar Constant variable

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

$$B = \begin{bmatrix} cos(\phi)dt & 0 \\ sin(\phi)dt & 0 \\ 0 & dt \\ 1 & 0 \end{bmatrix}$$

dt is a time interval.

5.9- Observation Model

The robot can get x-y position infomation from GPS.

- So GPS Observation model is Zt = H * xt
- H is constant by H = $\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$

5.10- Extented Kalman Filter equations in code

Localization process using Extendted Kalman Filter:EKF is

=== Predict ===

$$x_{Pred} = Fx_t + Bu_t$$

$$P_{Pred} = J_F P_t J_F^T + Q$$

=== Update ===

$$z_{Pred} = Hx_{Pred}$$

$$y = z - z_{Pred}$$

$$S = J_H P_{Pred.} J_H^T + R$$

$$K = P_{Pred.} J_H^T S^{-1}$$

$$x_{t+1} = x_{Pred} + Ky$$

$$P_{t+1} = (I - KJ_H)P_{Pred}$$

FIGURE7. 20: LOCALIZATION PROCESS USING KALMAN FILTER

Chapter 8

Path Planning

1- Introduction

Path planning is the task of finding a continuous path that will drive the car from the start to the end goal. The entire path must lie in the free space (as shown in Fig. 8.1). In path planning the car uses known environment map, which is stored in the memory.

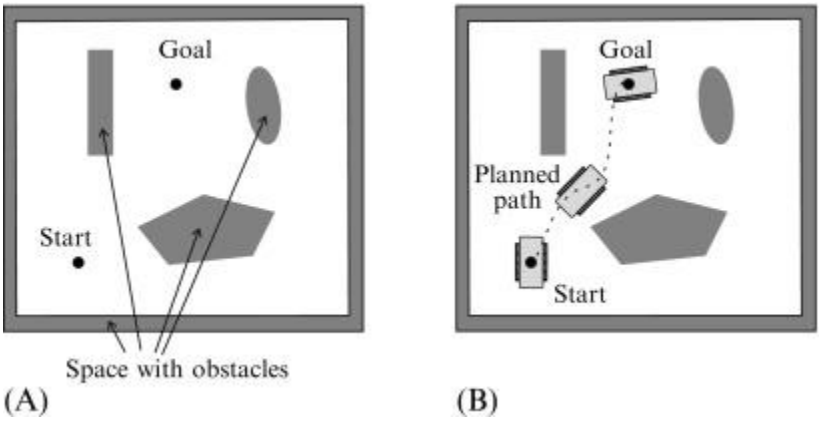


FIGURE8. 1: (A) ENVIRONMENT WITH OBSTACLES AND STARTING POINT AND GOAL CONFIGURATIONS, AND (B) ONE OUT OF MANY POSSIBLE PATHS FROM THE STARTING POINT TO THE GOAL CONFIGURATION

Path Planning uses data From Sensor fusion to understand the environment around use. And allows use data from localization to understand where we are in the environment. The Path Planning use all these data to construct a trajectory for the controller to execute.
The first step in being able to construct a path planning algorithm to construct a path from two points a start and an end is to know the environment in which the car well be put in. Since the car will be moving in the streets, we will need an algorithm that well be able to create a map of the real world that the car well be able to understand and well tell the car were the roads that it can drive on from the building and walking streets.

2- Map

So, we use **OpenStreetMap** or in short OSM. OSM creates free geographic data for the world streets. We choose OSM because most world and street maps you think of as free actually have legal or technical restrictions on their use.
OSM prove a figure of the of road, buildings, highways … etc (as shown in Fig. 8.2).

FIGURE8. 2: OSM MAP OF PART OF CITY

The problem with this map is that it is a contain un necessary data so first we need to remove all unnecessary data and leave only the roads. (as shown in Fig. 8.3).

FIGURE8. 3: OSM FOR THE SAME CITY BUT FOR STREETS ONLY

The second problem is that these is that OSM streets consists of edges and each edge consists of only two points and thought two point may be far apart. Since, we need to be able to know the states of the environut at every point we need to adjust the format of the map to be able to be used by the car.

FIGURE8. 4: OSM THAT WILL BE ABLE TO KNOW THE STATES OF THE ENVIRONMENT AT ANY POINT

Finally, this representation of the map (as shown in Fig. 8.4) well allow the car to know it's it environment at any point in the map.

3- Searching Algorithms

At its core, a pathfinding method searches a graph by starting at one point and exploring adjacent points until the destinations is reached, generally with the intent of finding the cheapest route. Although graph searching methods such as a breadth-first search would find a route if given enough time, other methods, which "explore" the graph, would tend to reach the destination sooner. An analogy would be a person walking across an environment rather than examining every possible route in advance, the person would generally walk in the direction of the goal and only diverge from the path to avoid an obstruction.
Two primary problems of pathfinding. The first is to find a path between two nodes in an environment. And the second is the shortest path problem which is to find the optimal shortest path. Basic algorithms such as breadth-first and depth-first search address the first problem by exhausting all possibilities that exist from starting from the given node, they iterate over all potential paths until they reach the destination node. These algorithms run linear time.
The more complicated problem is finding the optimal path. That are an exhaustive approach that search all the possibilities. However, it is not necessary to examine all possible paths to find the optimal one. Algorithms such as A* and Dijkstra's algorithm tactically eliminate paths, either by heuristics or through dynamic programming. By eliminating impossible paths.

3.1- Dijkstra's algorithm:

A common example of a graph-based pathfinding algorithm is Dijkstra's algorithm. This algorithm begins with a start point and an "open set" of candidate points. At each step, the point in the open set with the lowest distance from the starting point is examined. The node is marked "closed", and all nodes adjacent to it are added to the open set if they have not already been examined. This process repeats until a path to the goal has been found. Since the lowest distance nodes are examined first, the first time the goal is found, the path to it will be the shortest path.
So, it’s known that Dijkstra's algorithm are guaranteed to allows find the shortest path.

3.2- A* algorithm:

A* is a variant of Dijkstra's algorithm commonly used in games. A* assigns a weight to each open point equal to the weight of the distances to that point plus the approximate distance between that point and the goal. This approximate distance is found by the heuristic function. And represents a minimum possible distance between that point and the end. This allows it to eliminate longer paths once an initial path is found. If there is a path of length x between the start and finish, and the minimum distance between a point and the finish is greater than x, that point need not be examined. A* is although known to faster than Dijkstra's algorithm

3.3- Why A*?

Scene the searching algorithm well be done repeatedly and we need the searching algorithm to be fast and the A* is known the fast searcher.

4- What is heuristic function?

The heuristic function h(n) tells A* an *approximation* of the minimum cost from any point to the goal. It's important to choose a good heuristic function.
The heuristic can be used to control A*'s behavior:

- At one extreme, if h(n) is 0, then only g(n) (the distance between any point and start) plays a role, and A* turns into Dijkstra's Algorithm, which is certain to find a optimal path.
- If h(n) is always lower than (or equal to) the cost of moving from n to the goal, then A* is guaranteed to find a shortest path. The lower h(n) is, the more points A* expands, making it slower.
- If h(n) is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.
- If h(n) is sometimes greater than the cost of moving from n to the goal, then A* is not certain to find a shortest path, but it can run faster.
- At the other extreme, if h(n) is very high relative to g(n), then only h(n) plays a role, and A* turns into Greedy Best-First-Search.

So, we have an interesting problem in that we can decide what we want to get out of A*. With 100% accurate estimates, we'll get shortest paths really quickly. If we're too low, then we'll continue to get shortest paths, but it'll slow down. If we're too high, then we give up shortest paths, but A* will run faster.
On a map, there are well-known heuristic functions to use.

Use the distance heuristic that matches the allowed movement:

- On a square grid that allows **4 directions** of movement, use the Manhattan distance.
- On a square grid that allows **8 directions** of movement, use the Diagonal distance.
- On a square grid that allows **any direction** of movement, you might or might not want to use Euclidean distance.

4.1- Manhattan distance:

The standard heuristic for a square grid is the Manhattan distance

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return  (dx + dy)
```

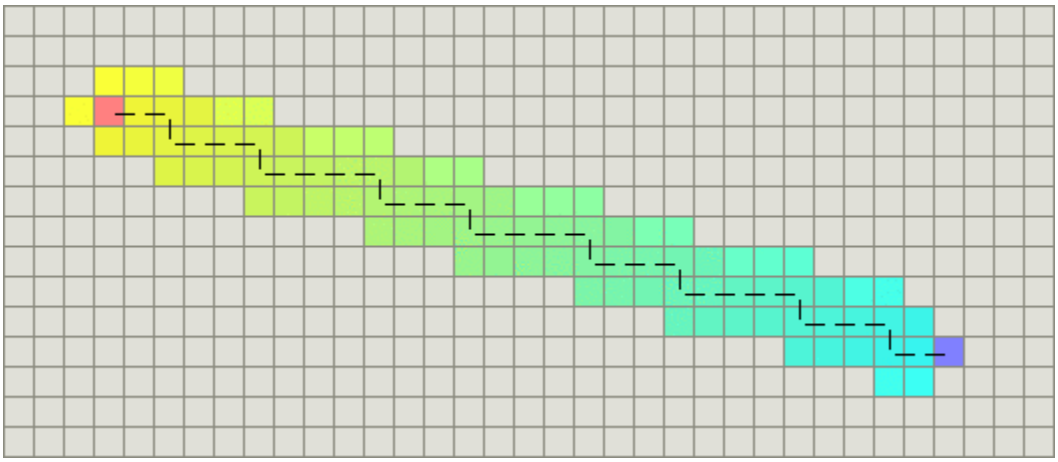


FIGURE8. 5: MANHATTAN DISTANCE

4.2- Diagonal distance:

If your map allows diagonal movement you need a different heuristic.

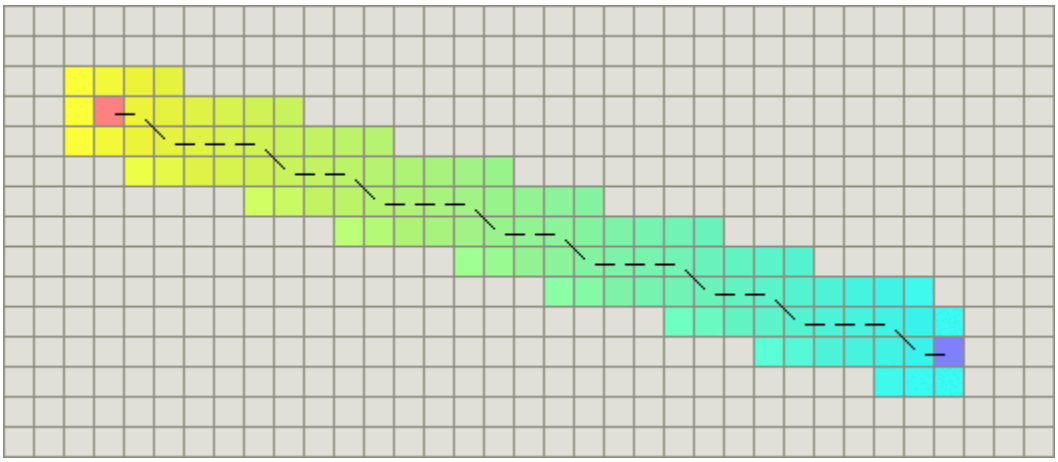


FIGURE8. 6: DIAGONAL DISTANCE

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy) + (D2 - 2 * D) * min(dx, dy)
```

where D is the cost of moving up or down right or left

where D2 is the cost of moving diagonally

4.3- Euclidean distance:

If your units can move at any angle (instead of grid directions), then you should probably use a straight-line distance:

```
Function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    Return sqrt(dx * dx + dy * dy)
```

However, if this is the case, then you may have trouble with using A* directly because the cost function `g` will not match the heuristic function `h`. Since Euclidean distance is shorter than Manhattan or diagonal distance, you will still get shortest paths, but A* will take longer to run:

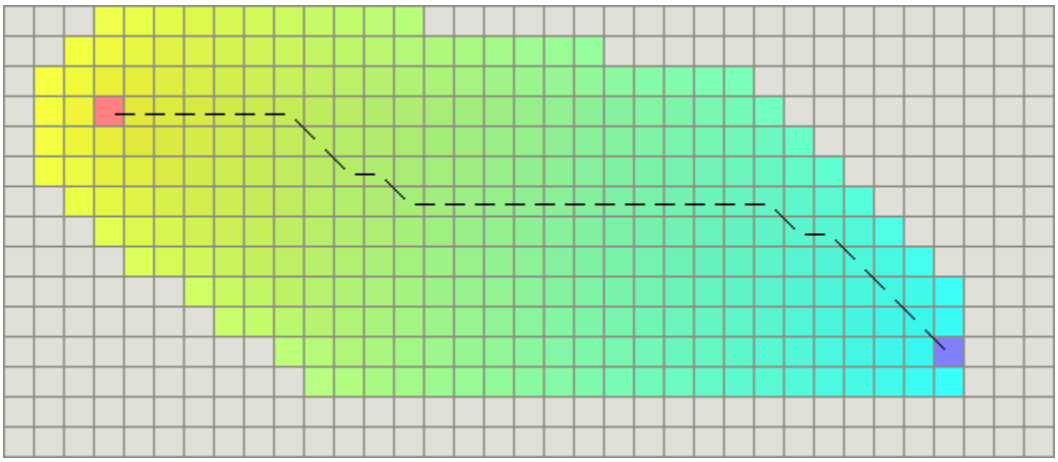


FIGURE8. 7: EUCLIDEAN DISTANCE

4.4- What is the problems of A* searching algorithm?

In some times A* can produce a path that the car is in capable of executing like a ninety degree turn the car cannot rotate ninety degrees on its axis it needs to rotate in an arc.
So, the solution is to use A*hybrid

4.5- A*hybrid:

A*hybrid take the steering angle and the velocity of the car and expand in the grid using the model of the car.
In this way the path that the A*hybrid will produce will be guaranteed to be executable bay the car

4.6- Conclusion:

Know we are able to create a map of the real environment and we have a searching algorithm that is capable of producing a path that the car is capable to follow.

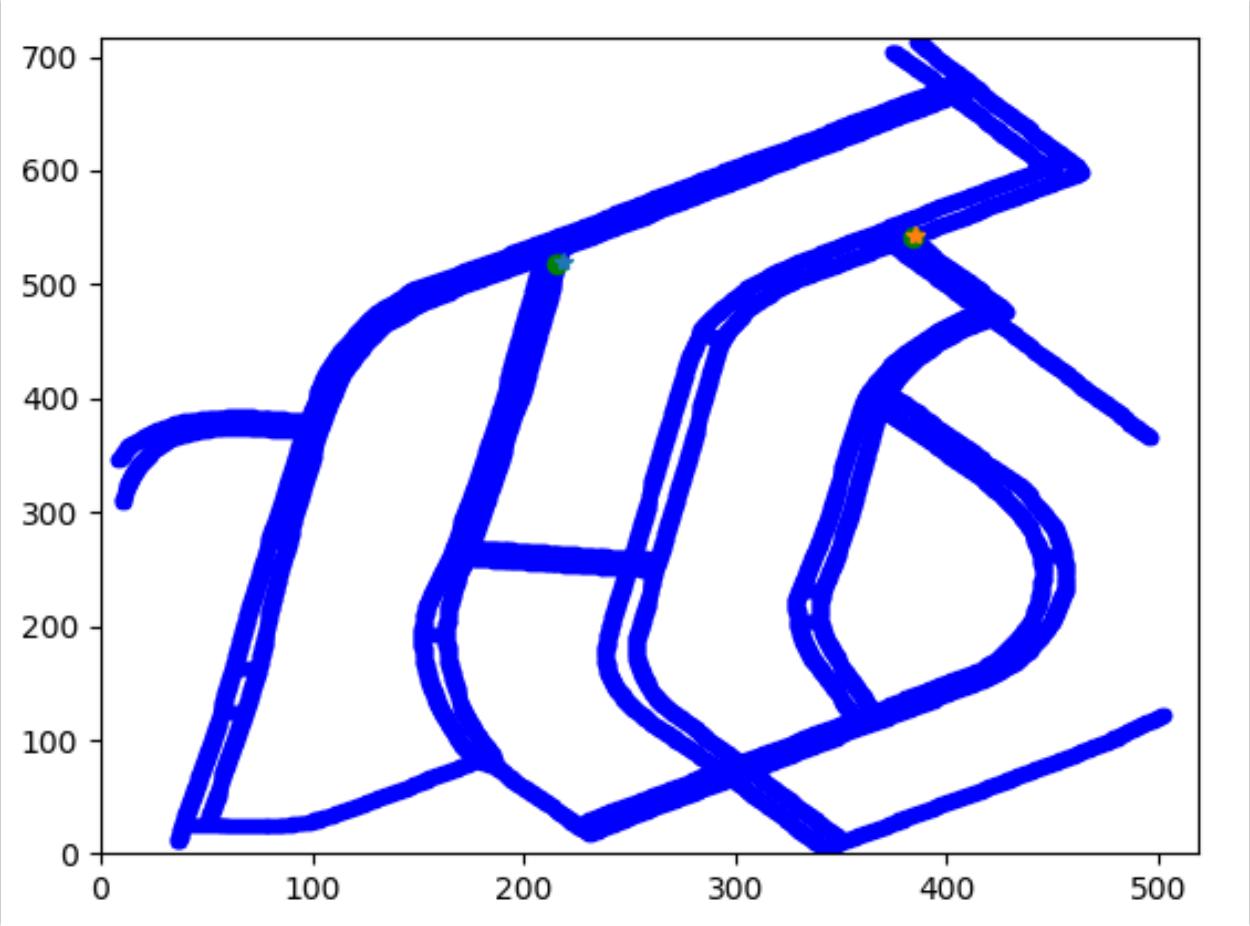


FIGURE8. 8: MAP IN WITH TO SEARCH

In this map we are trying to produce a path from start green star to the goal orange star

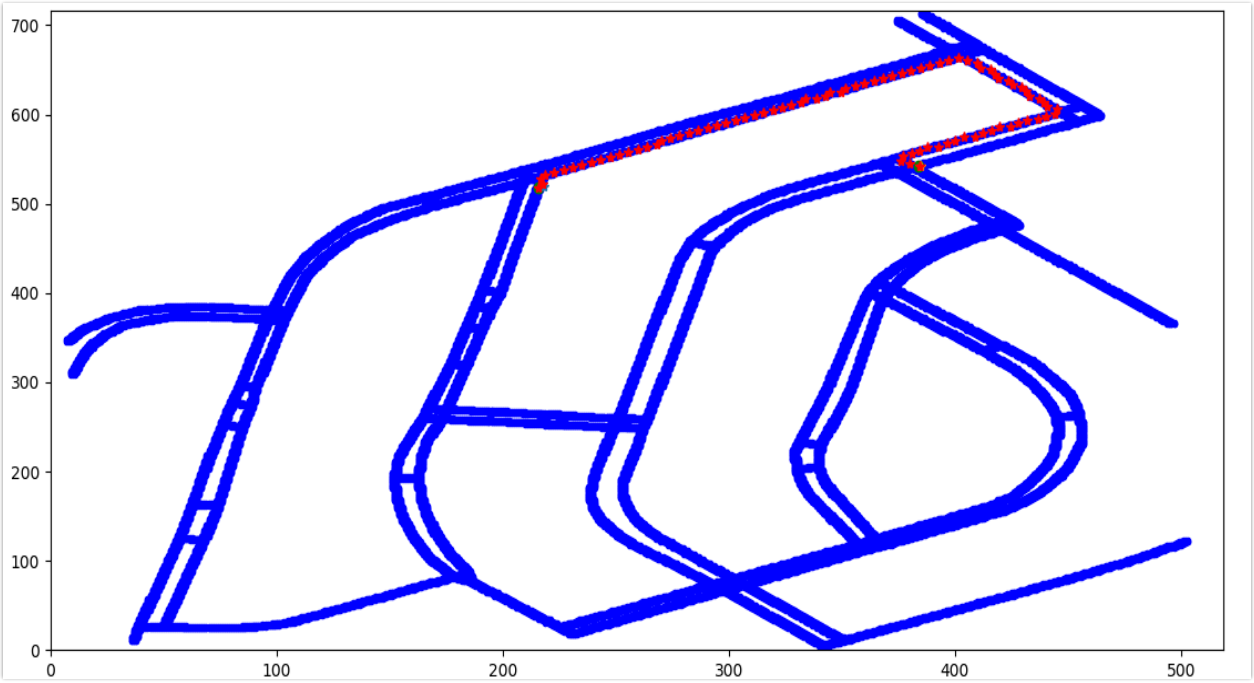


FIGURE8. 9: PATH THAT THE CAR WILL FOLLOW

After applying the A*hybrid algorithm we are able to produce a global path that the car well follow

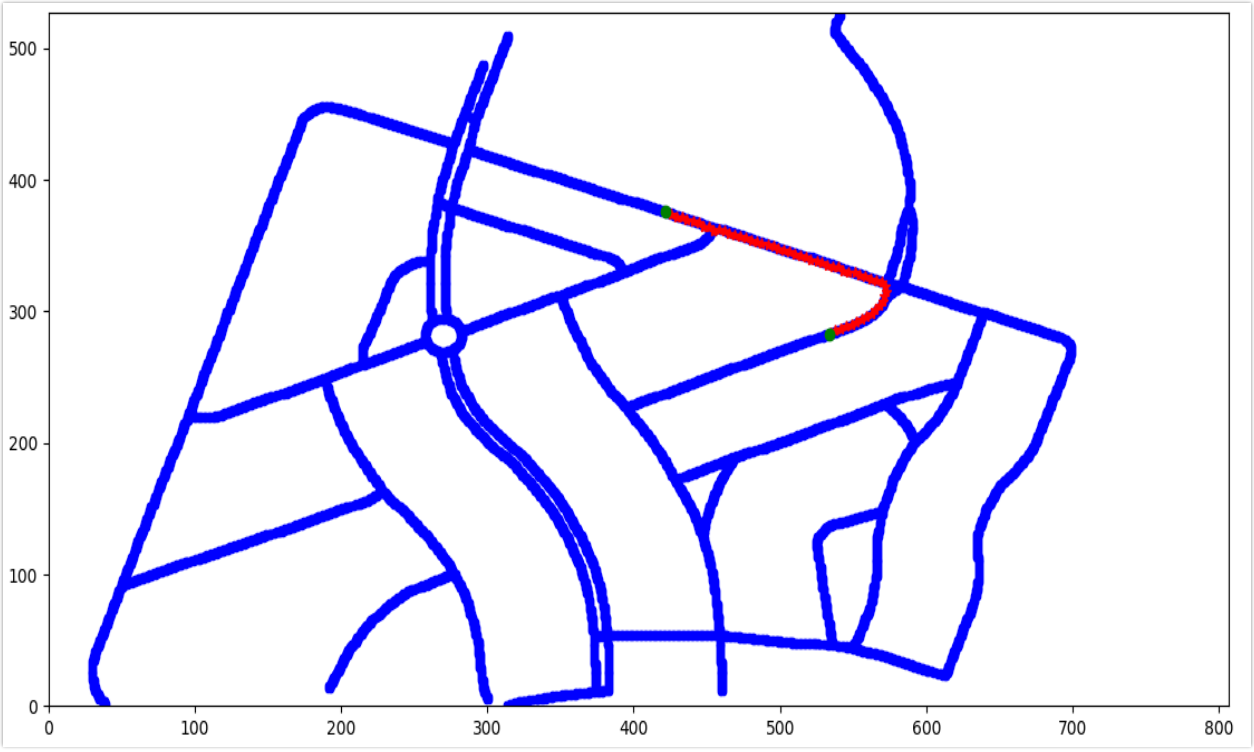


FIGURE8. 10: ANOTHER EXAMPLE TO THE PATH PLANNING

Chapter 9

Computer Vision

1- Introduction

Computer-vision help autonomous vehicles to perceive the environment and make the optimum decision, it is one of the most important components of self-driving cars.
Lane finding, road segmentation, object detection and localization, traffic sign recognition, traffic light recognition and much more, all of this based on computer vision.
In our project we will introduce and implement two features, Lane finding and road segmentation.

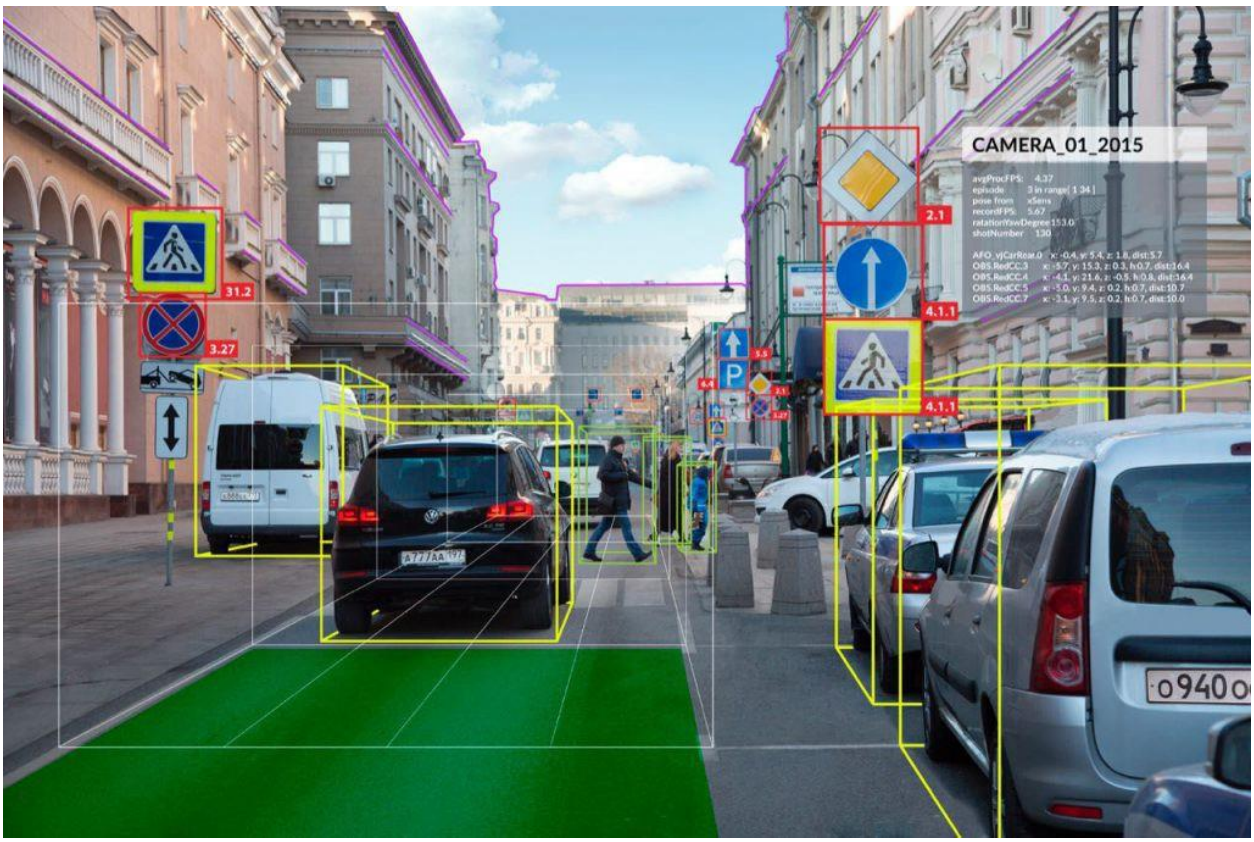


FIGURE9. 1: LANE FINDING AND ROAD SEGMENTATION

2- Lane finding

Lane finding is the task to find the area between the two lanes lines in the road or highways.

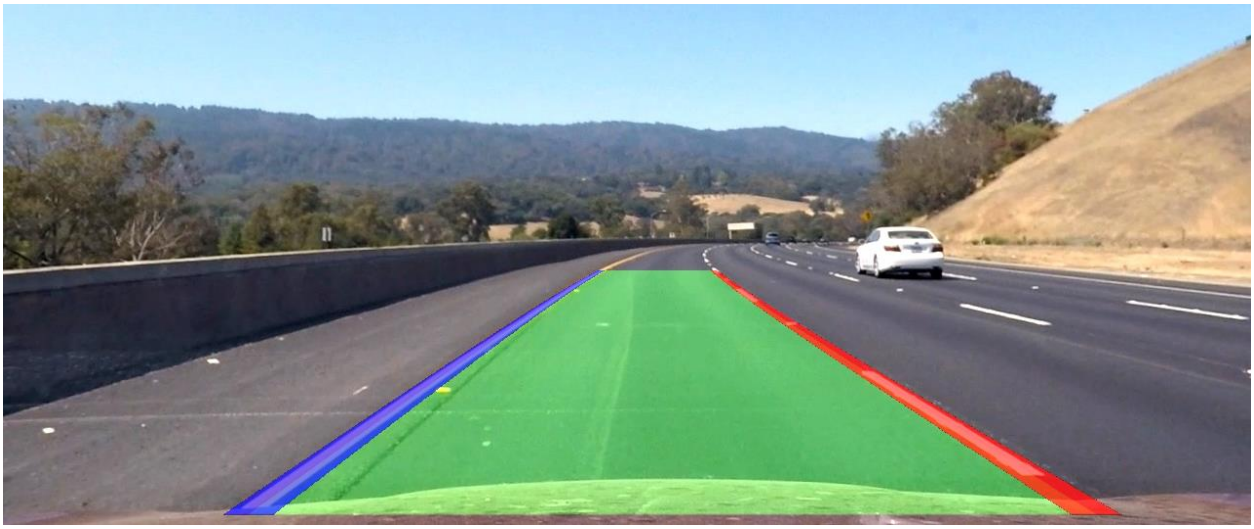


FIGURE9. 2: LANE FINDING

We used python programming language and intel open-cv library to perform the image operation.

First, we start with image preprocessing, One way to separate and detect objects in an image is to use color transforms like (RGB, HLS) and gradients to generate a filtered-down thresholder binary image.

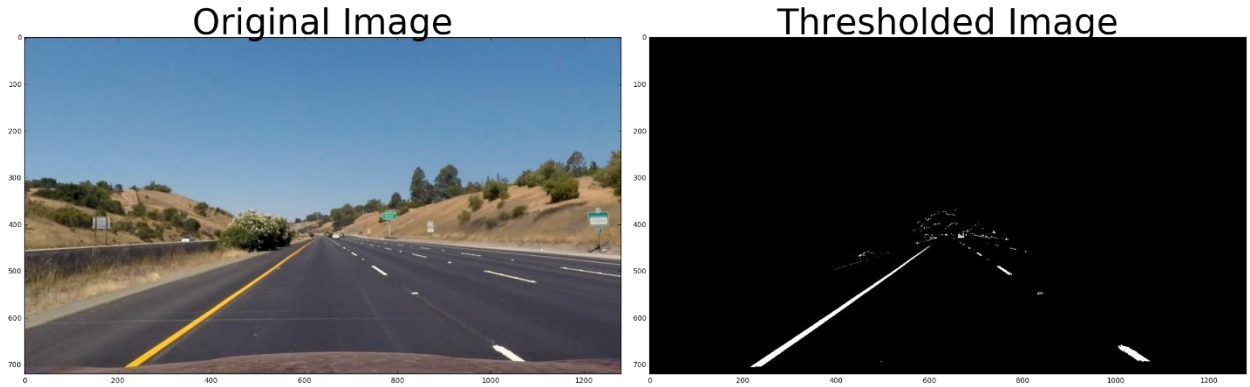


FIGURE9. 3 : IMAGE PROCESSING COMPARSION

Second, perspective transform ("birds-eye view"). We manually examining some of the sample images, we extracted the vertices to perform a perspective transform. The polygon with these vertices is drawn on the image for visualization. Destination points are chosen such that straight lanes appear more or less parallel in the transformed image.

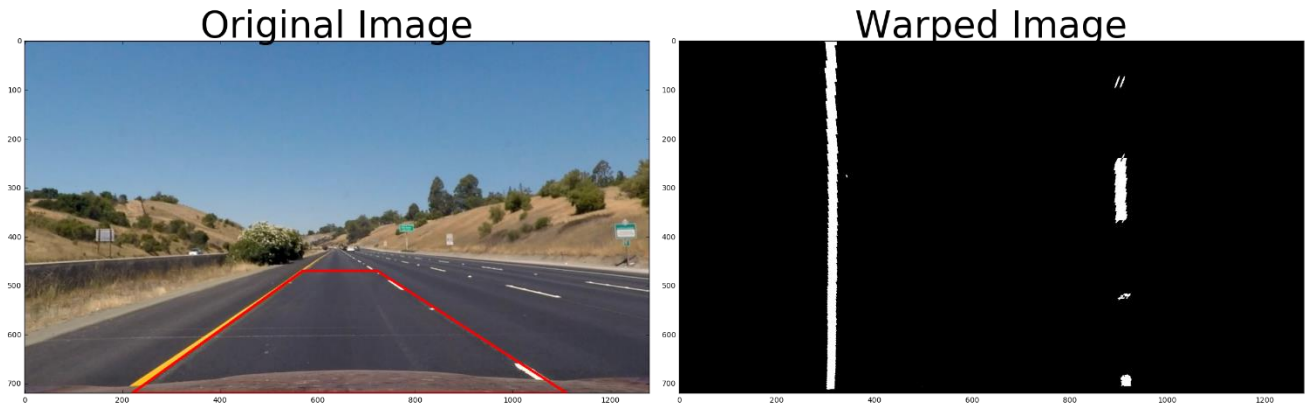


FIGURE9. 4: IMAGE PROCESSING ORIGINAL AND WRAPED

Third, Detect lane pixels (sliding window search). we perform a sliding window search method, starting with the base likely positions of the 2 lanes, calculated from the histogram. we have used 10 windows of width 100 pixels.
The x & y coordinates of non zeros pixels are found, a polynomial is fit for these coordinates and the lane lines are drawn.

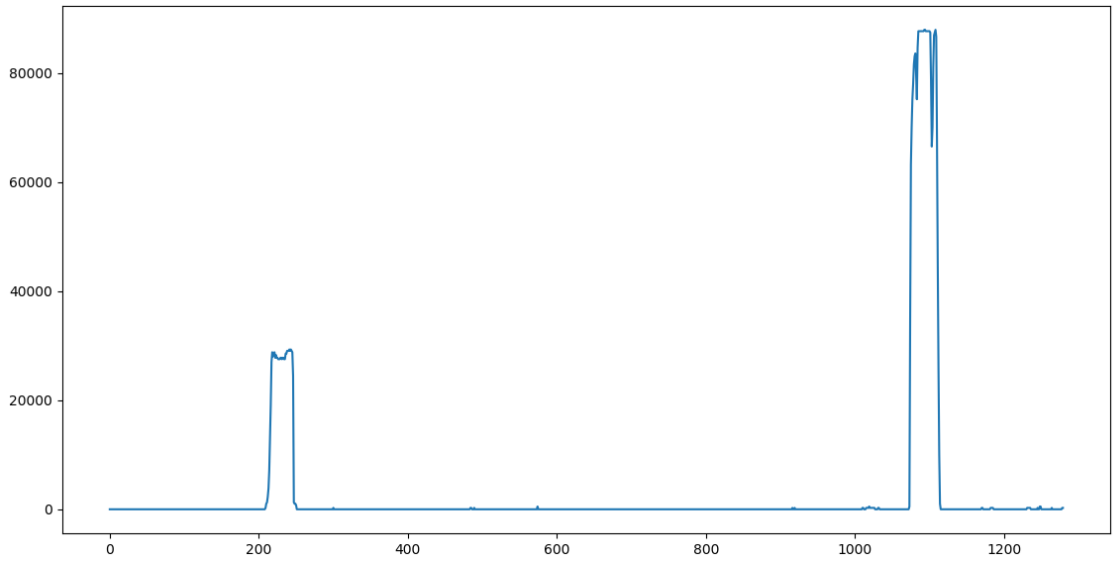


FIGURE9. 5: HISTOGRAM

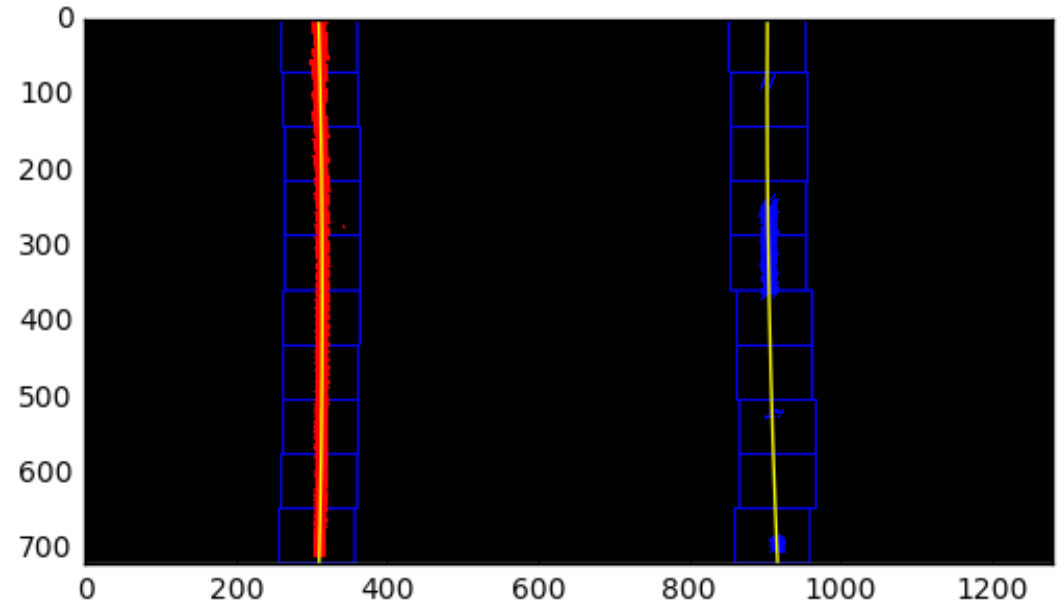


FIGURE9. 6: BOOKS OF WINDOW SEARCH

Fourth Searching around previously detected lane line Since consecutive frames are likely to have lane lines in roughly similar positions, we search around a margin of 50 pixels of the previously detected lane lines.

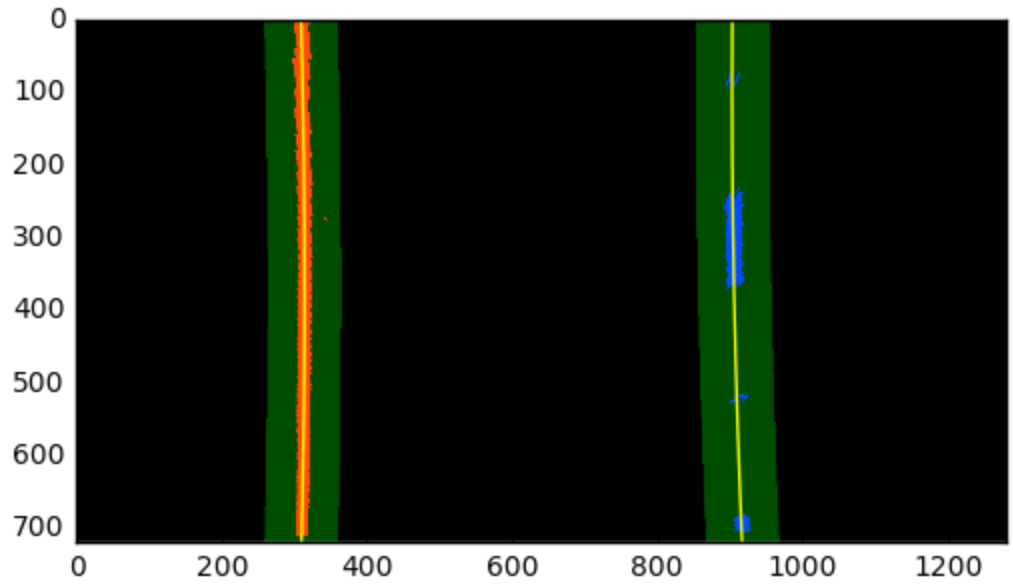


FIGURE9. 7: WINDOW SEARCH

Finley Inverse transform and output For the final image we:

- Paint the lane area
- Perform an inverse perspective transform
- Combine the processed image with the original image.

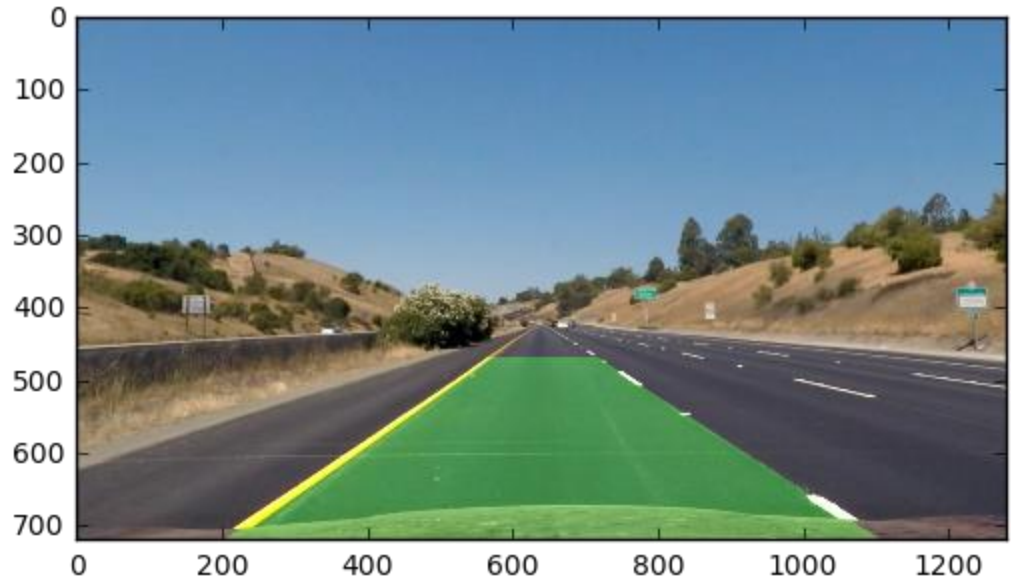
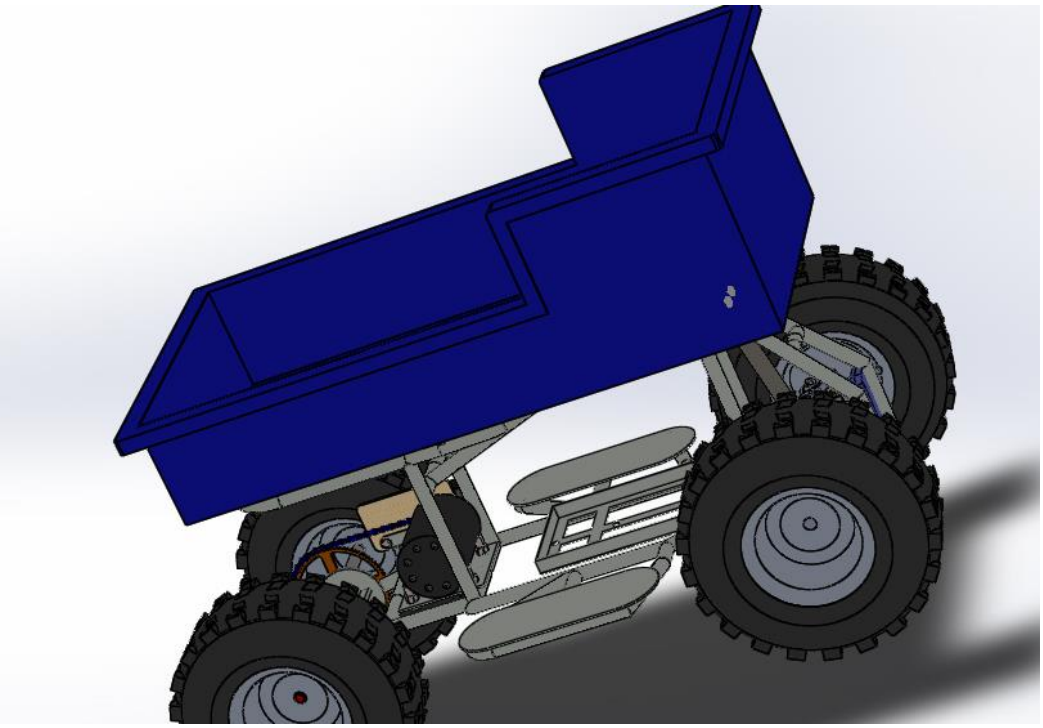


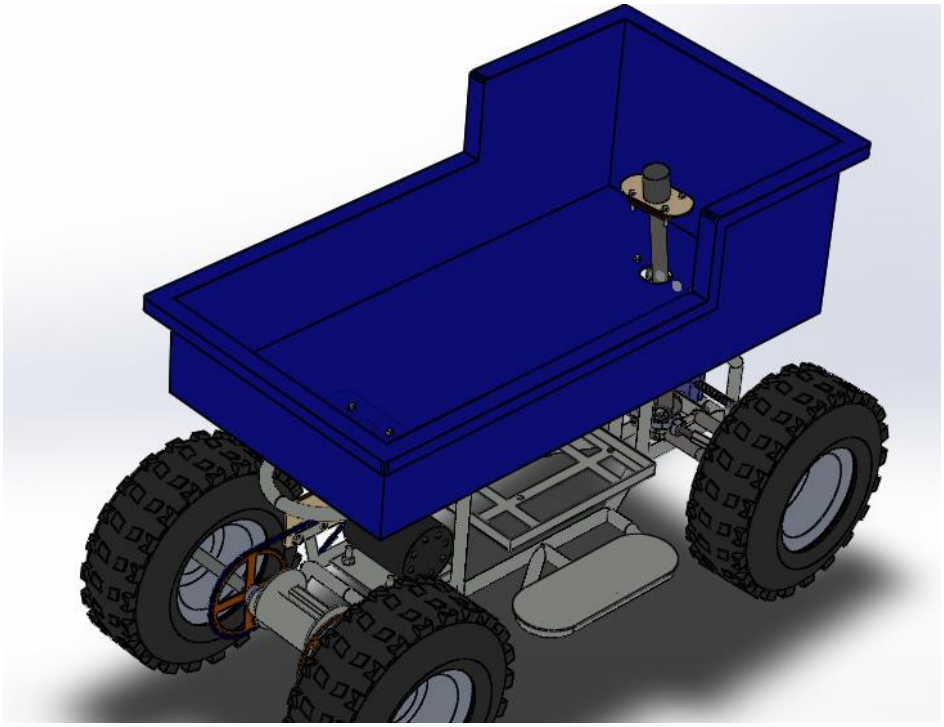
FIGURE9. 8: LANE KEEPING

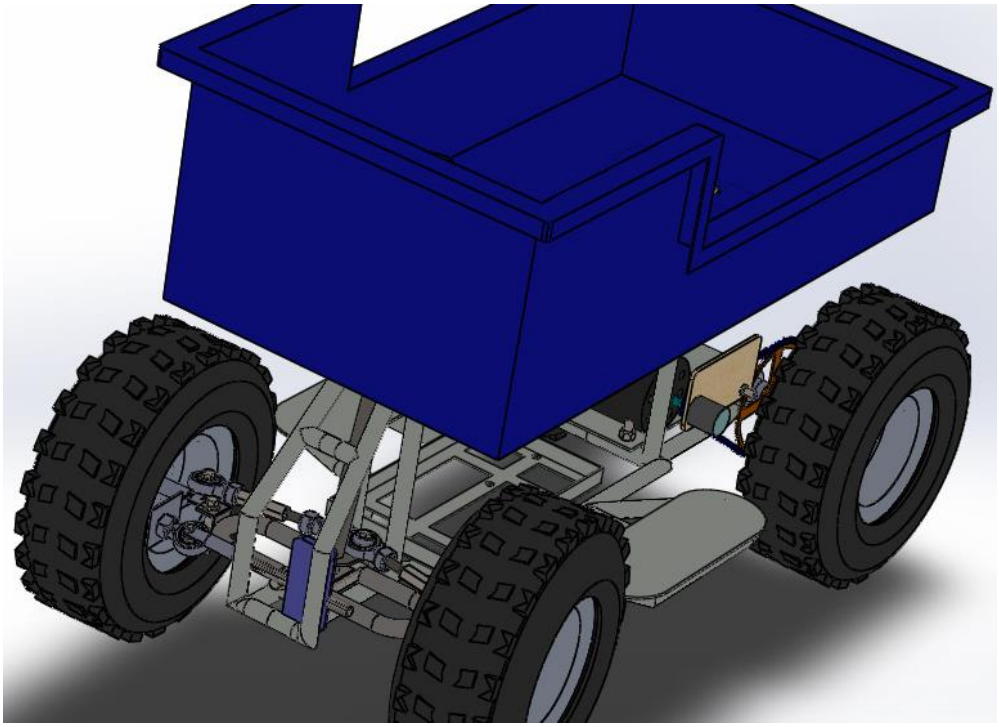
Appendix

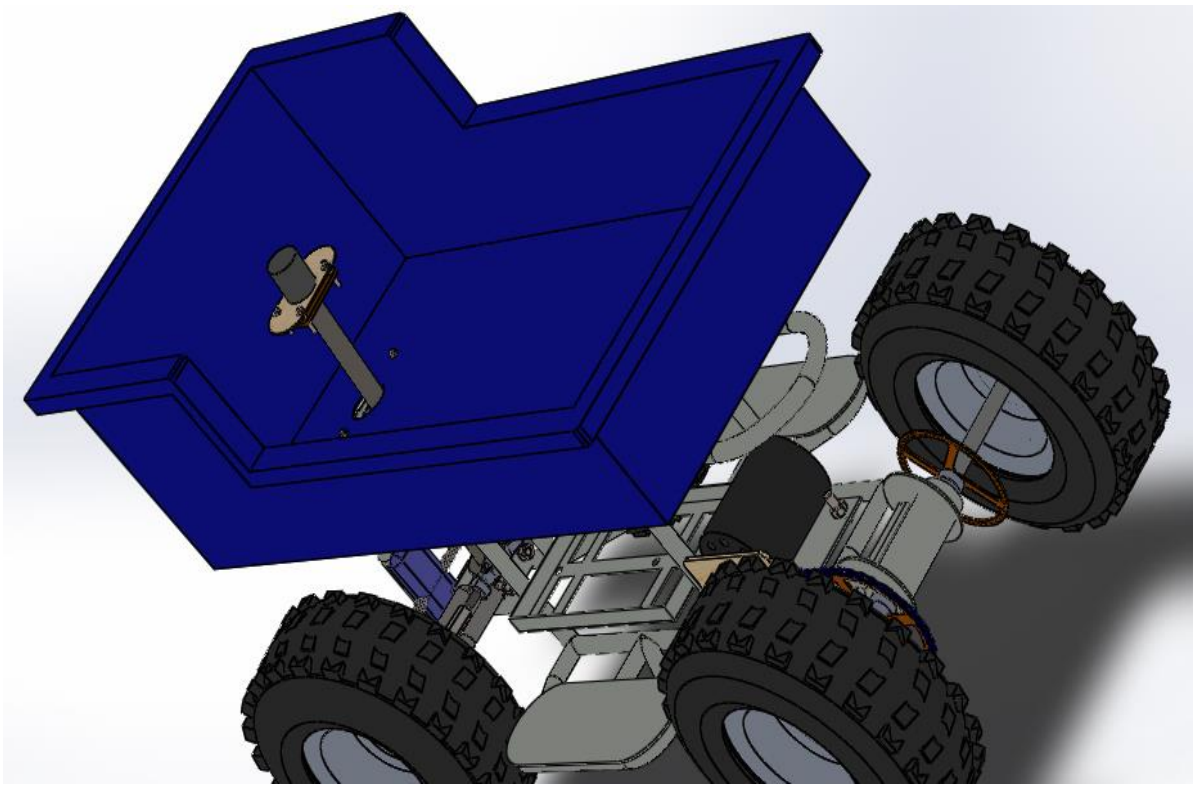
Components of robots:

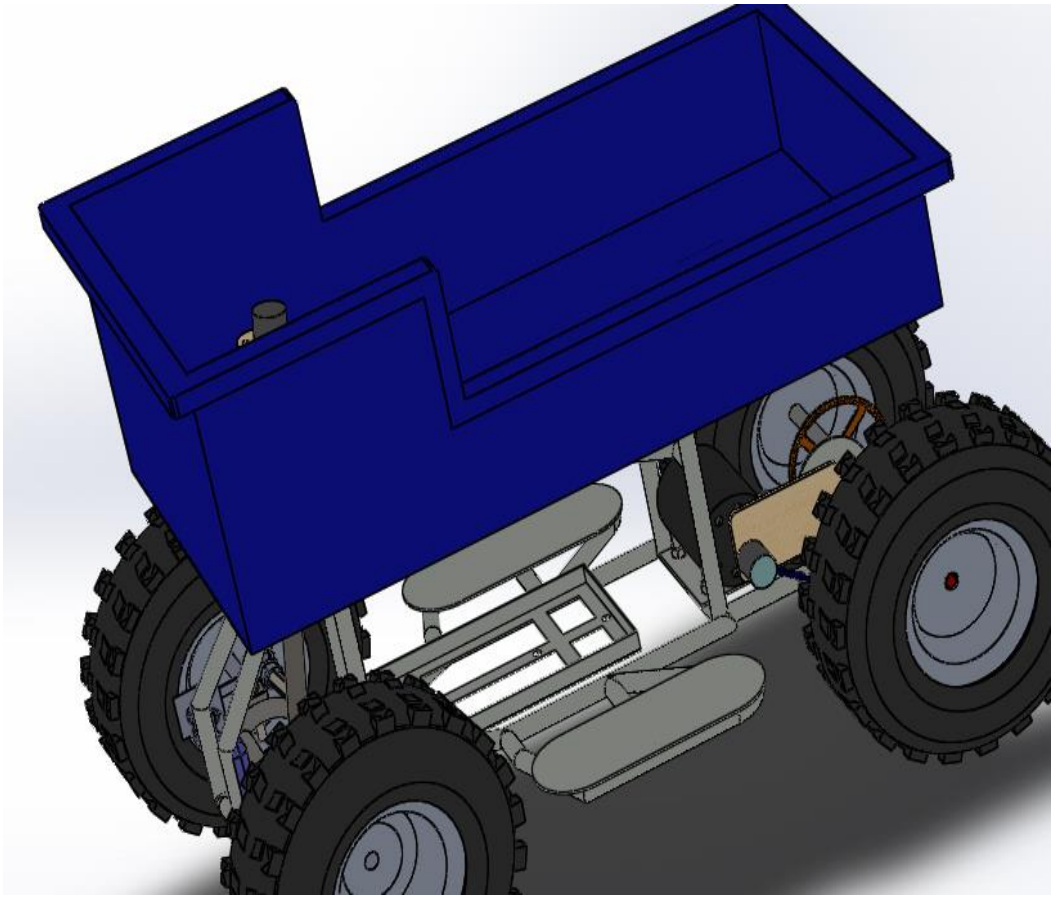
Final design:



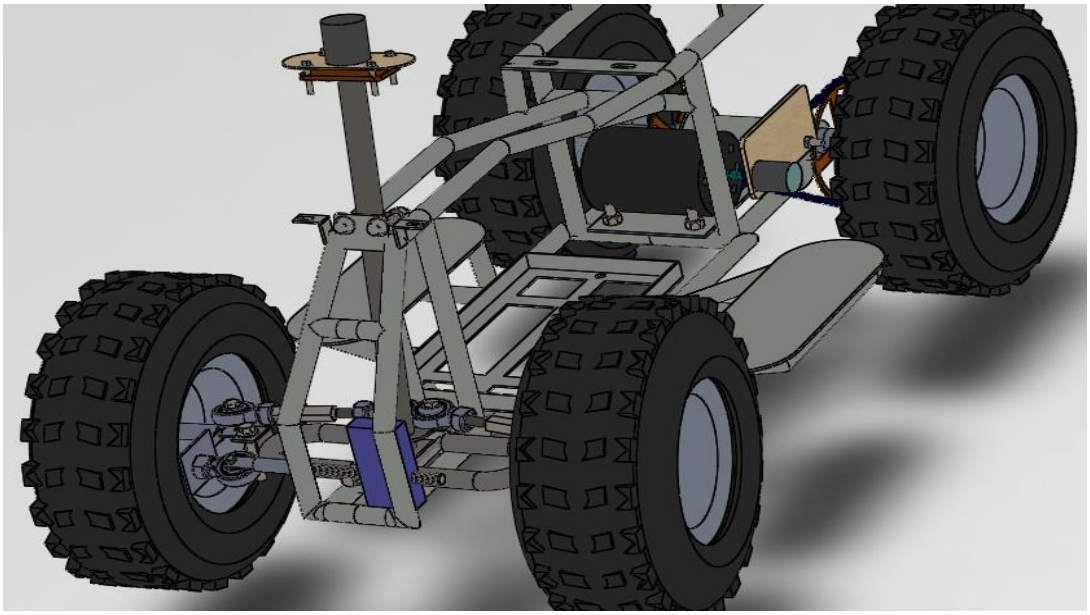


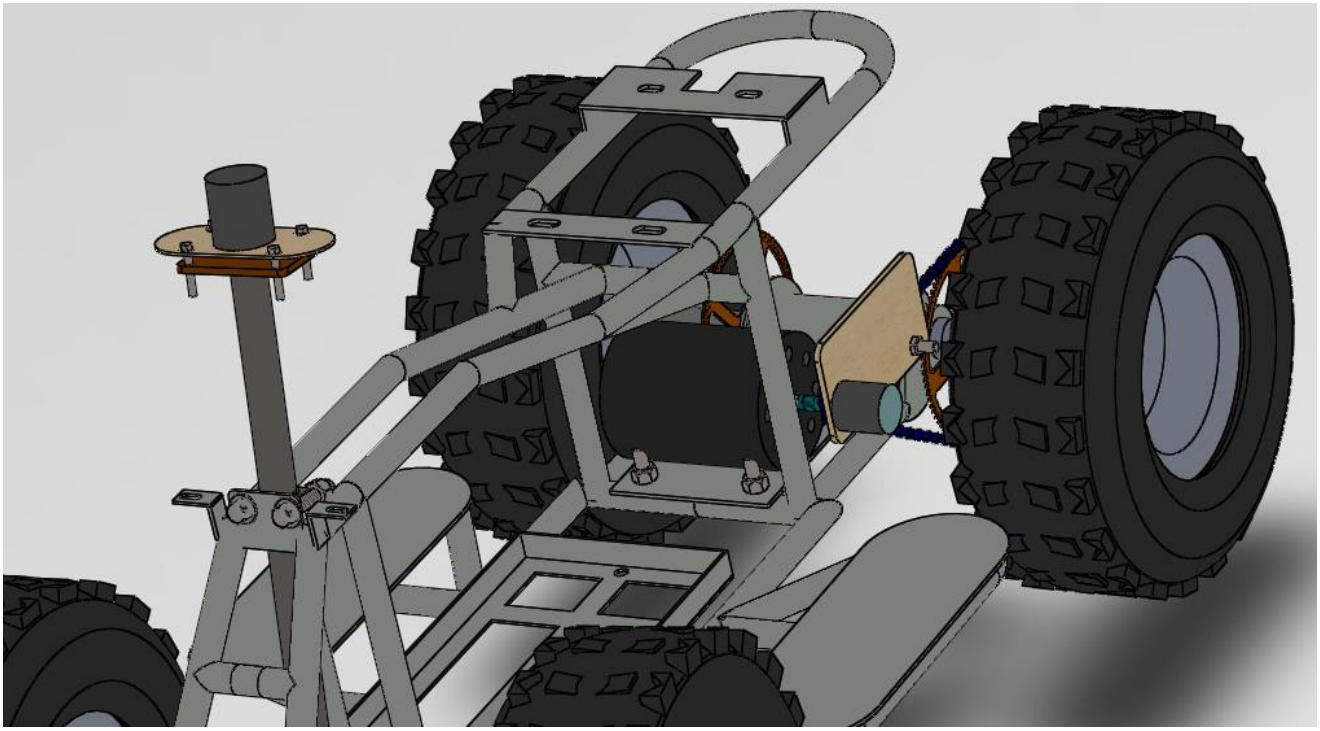


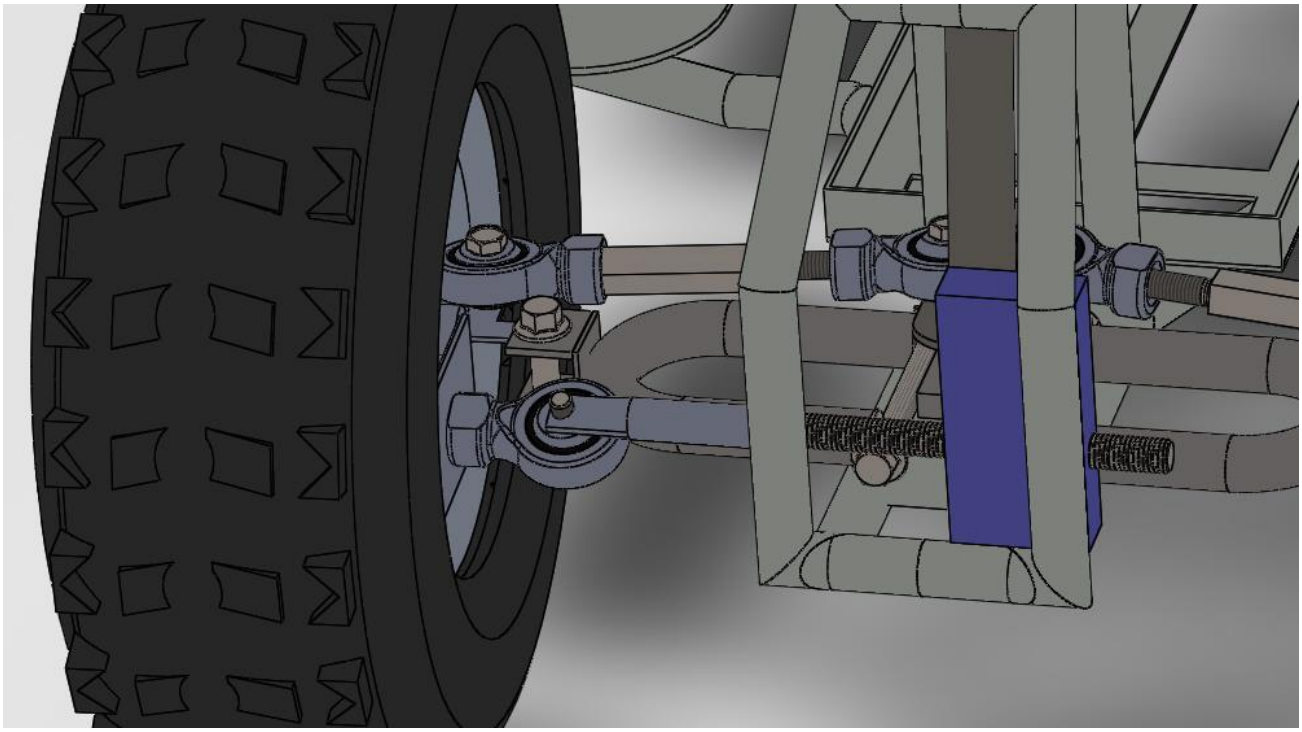


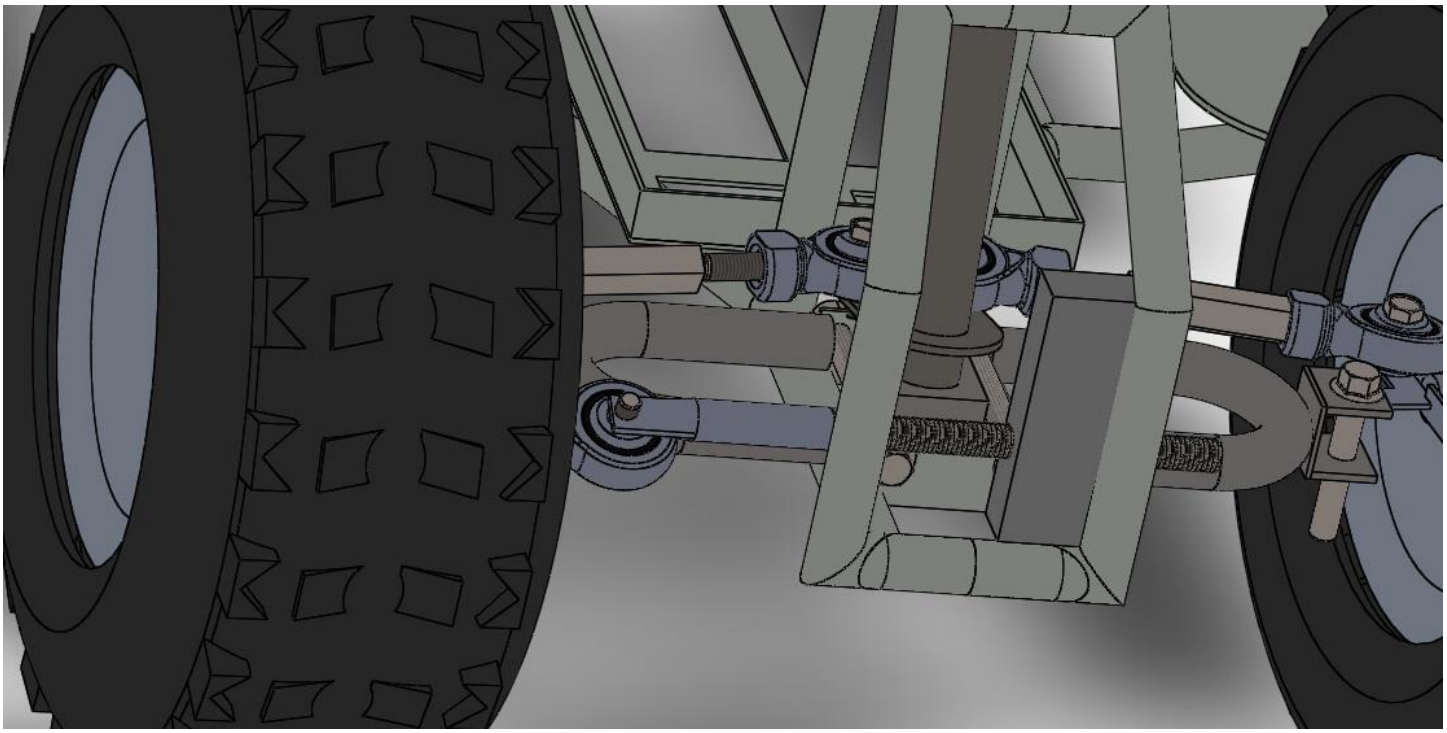


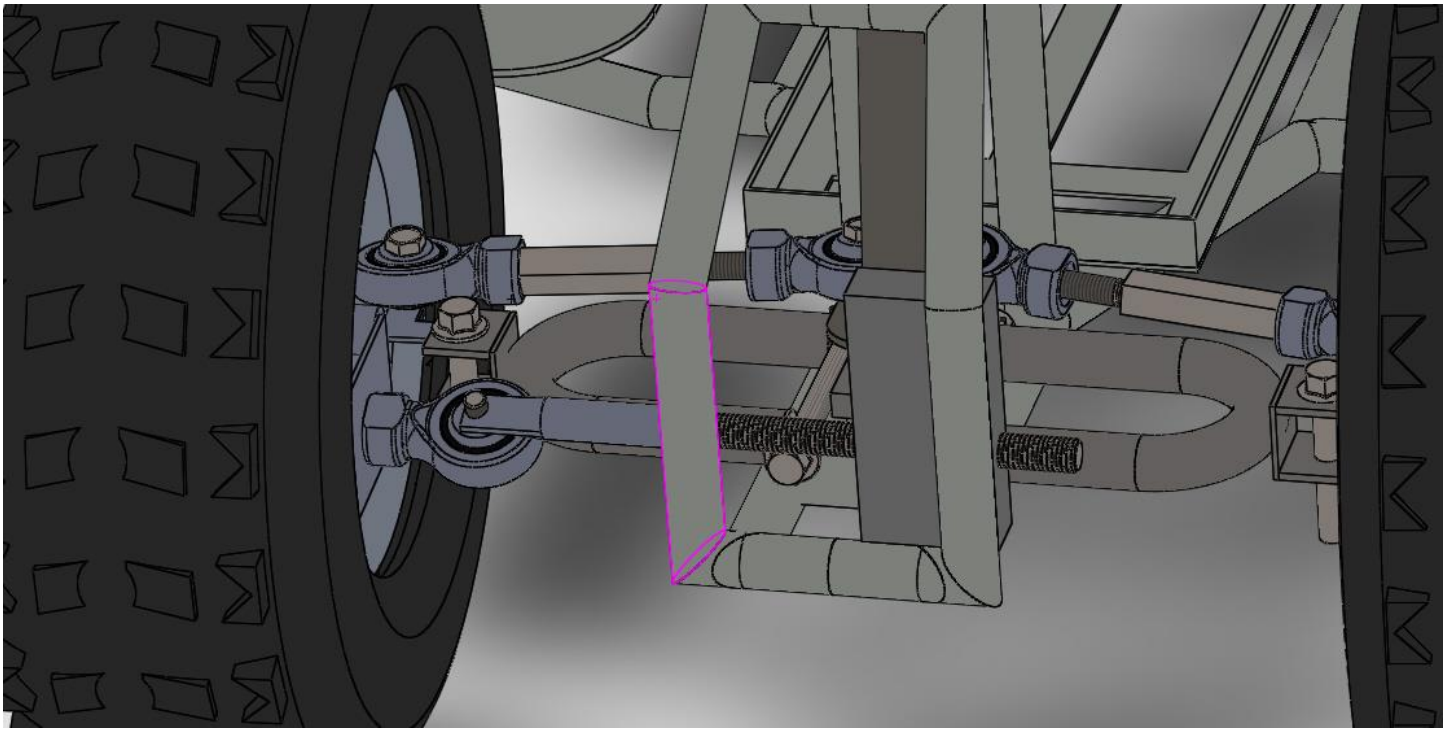
Final design without box:

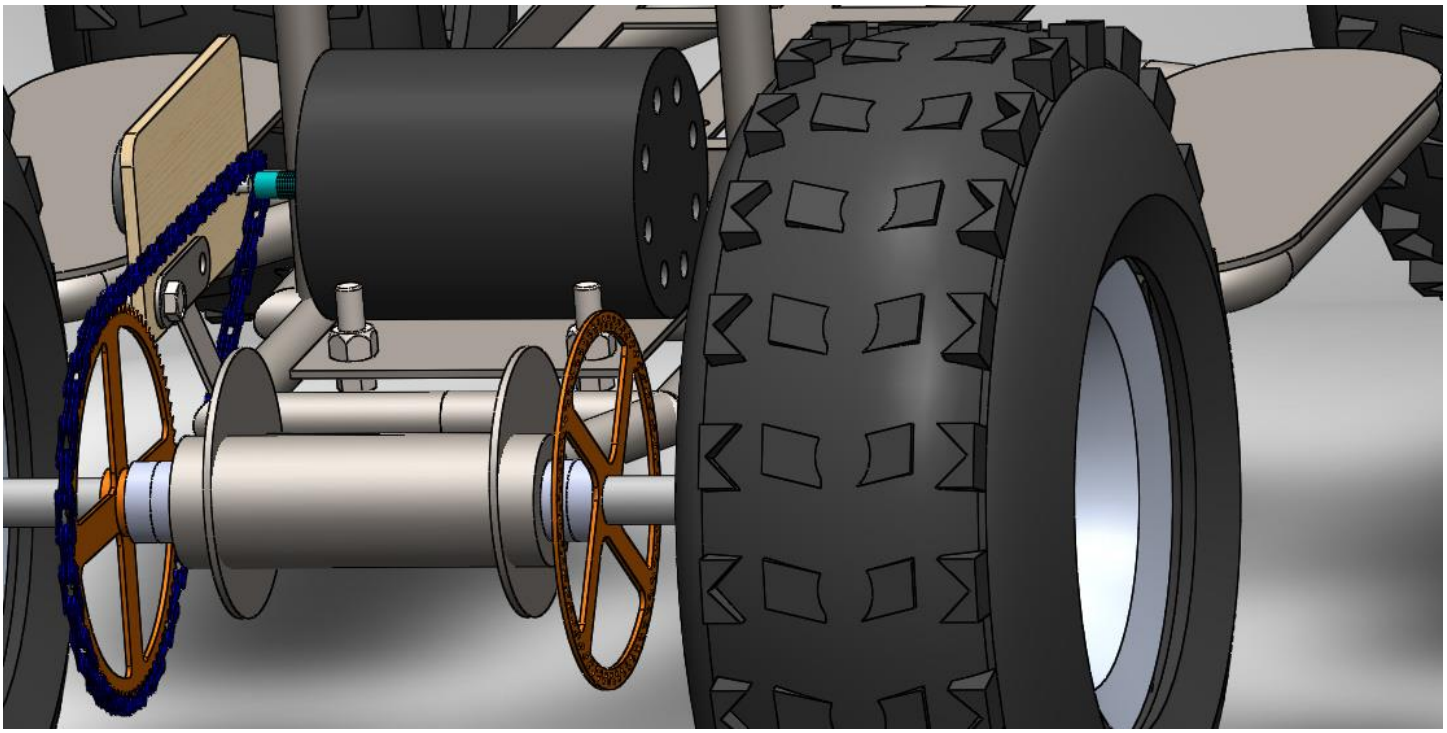


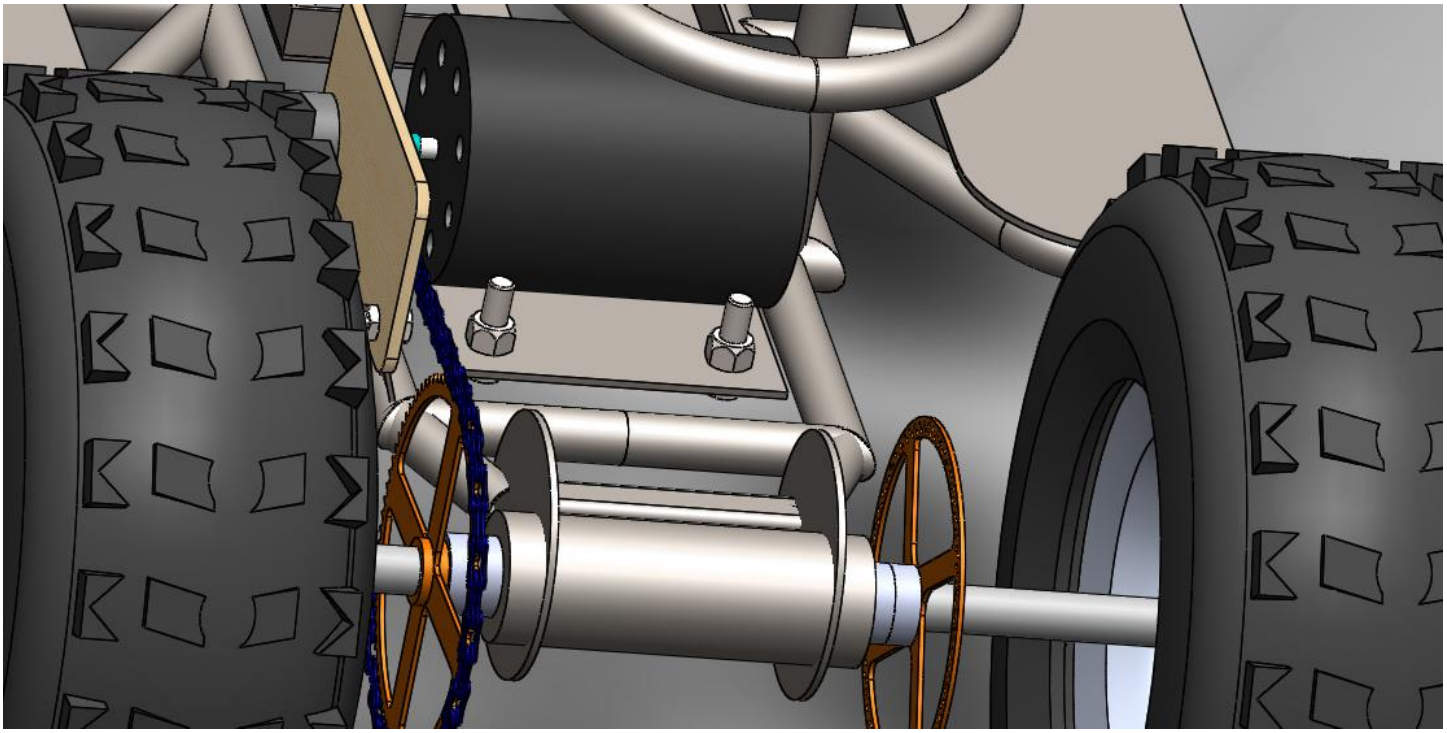


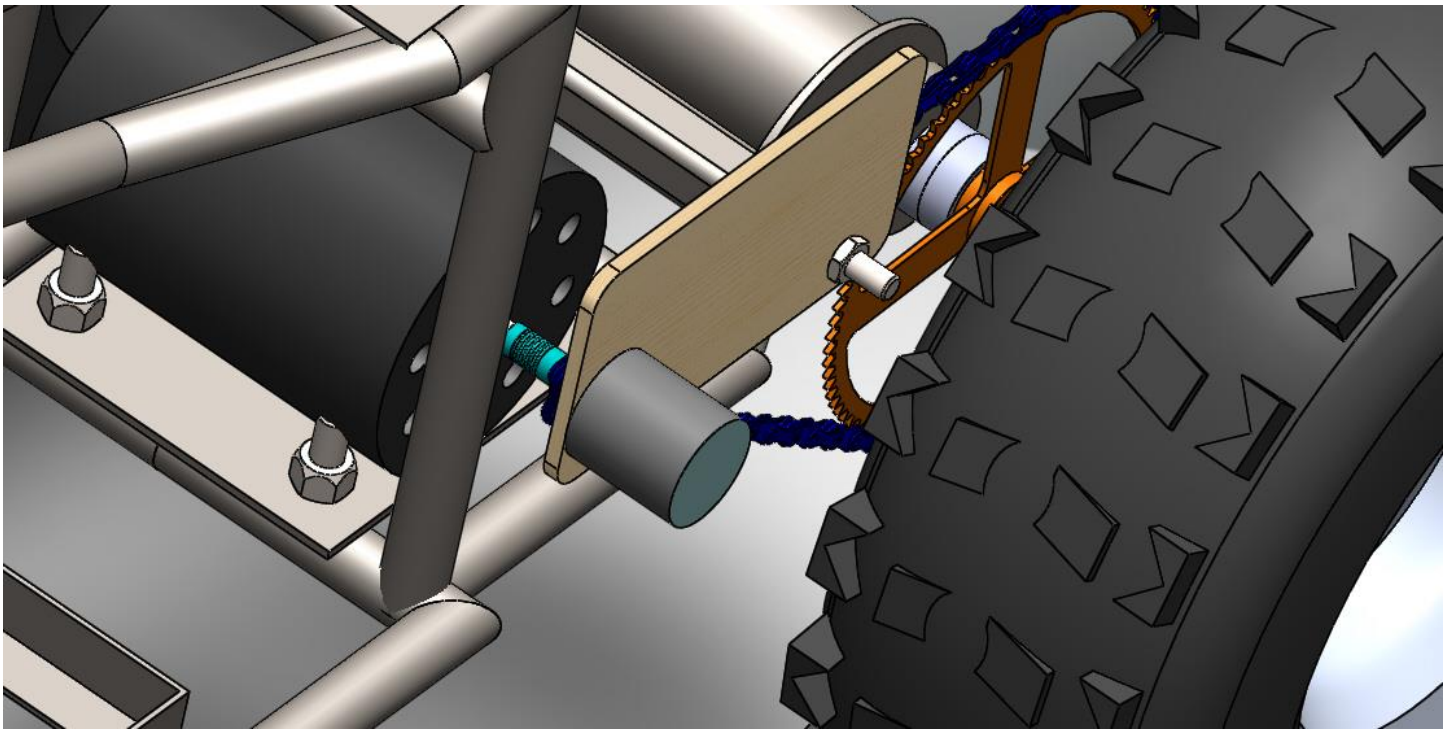


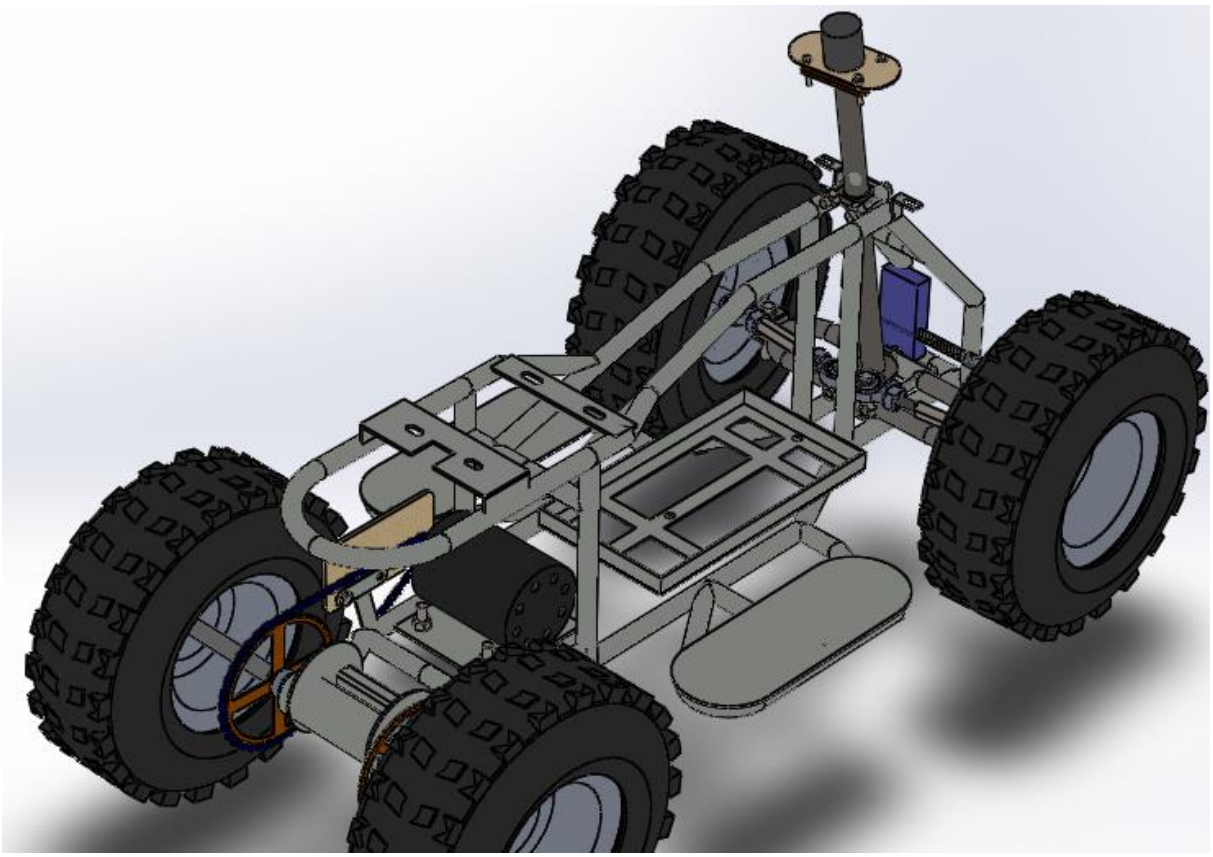


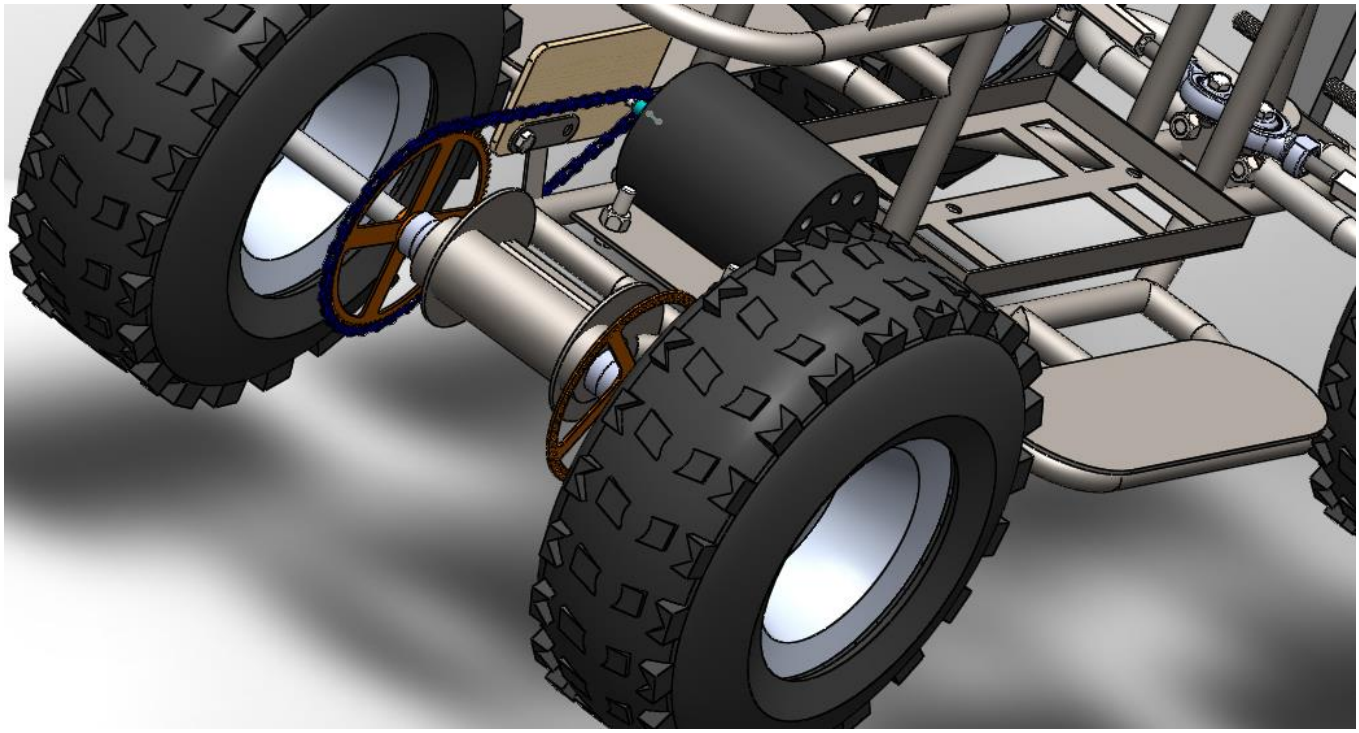




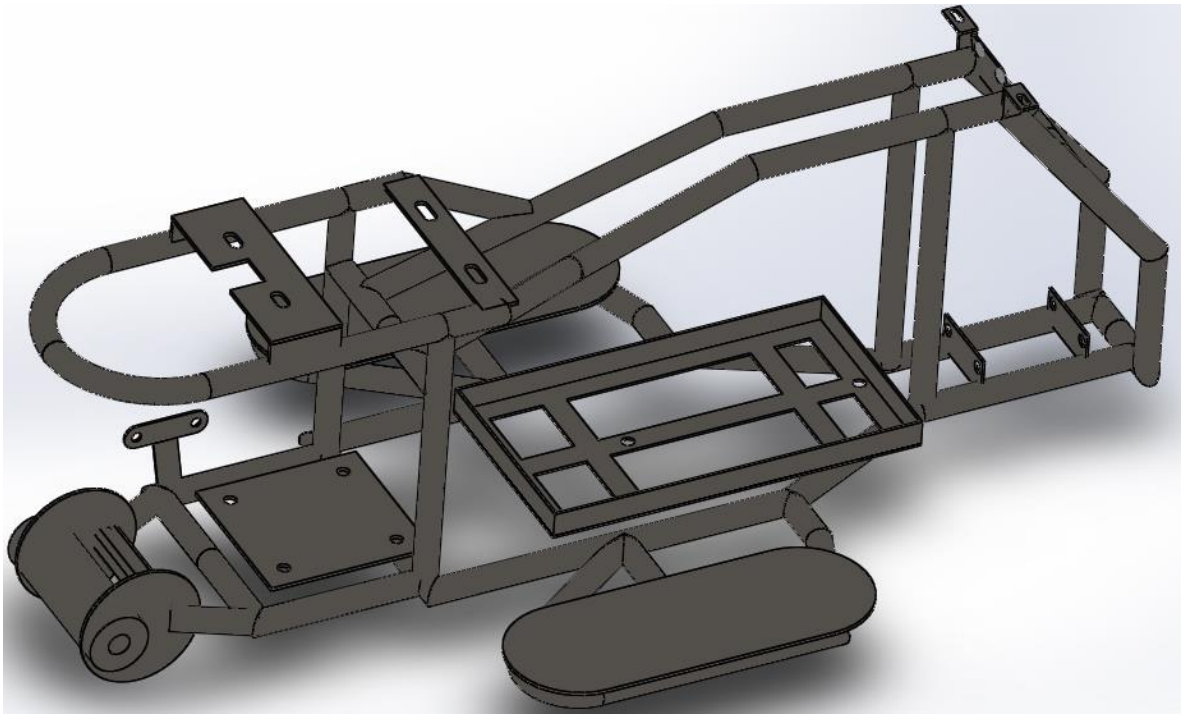


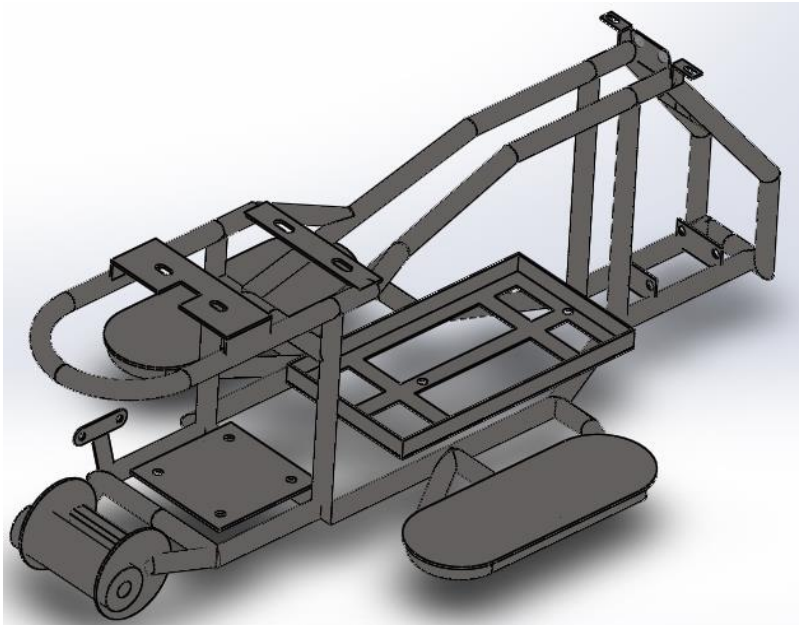




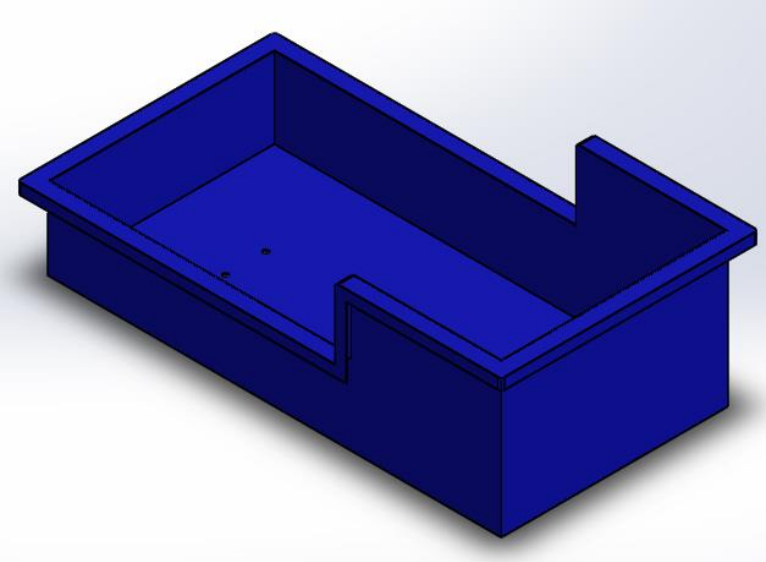


Base of robot:

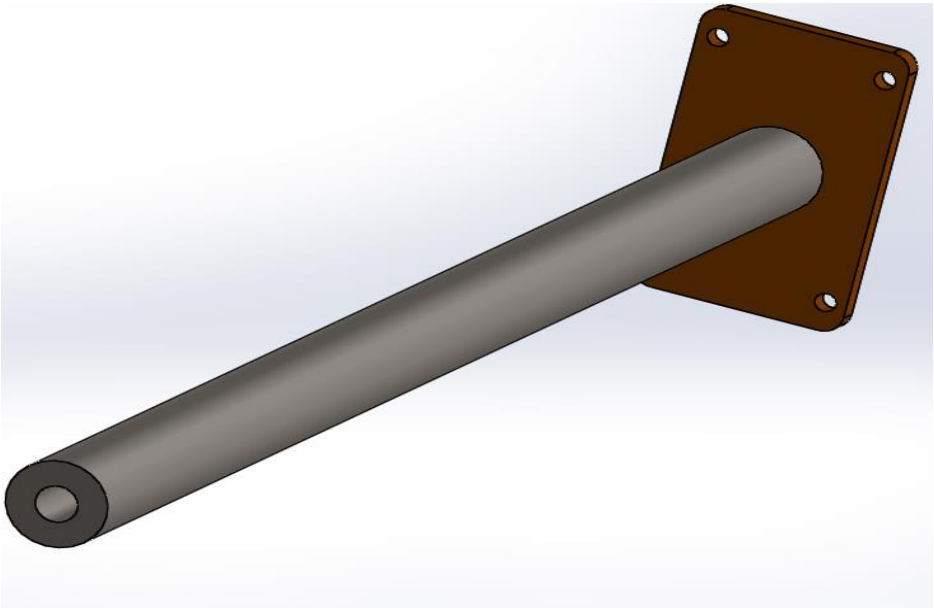


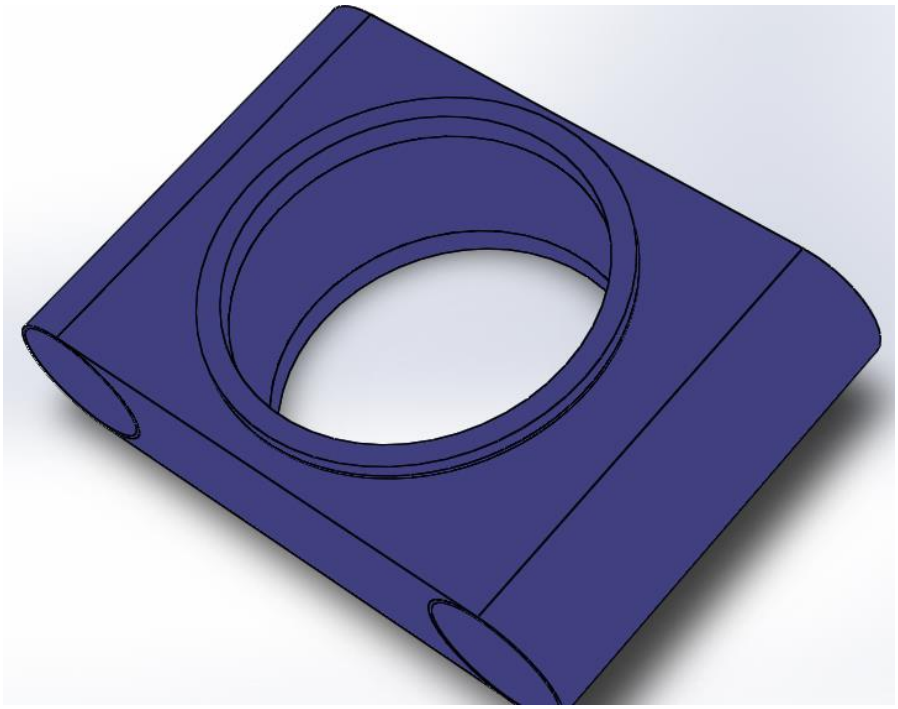


Box for compounds:

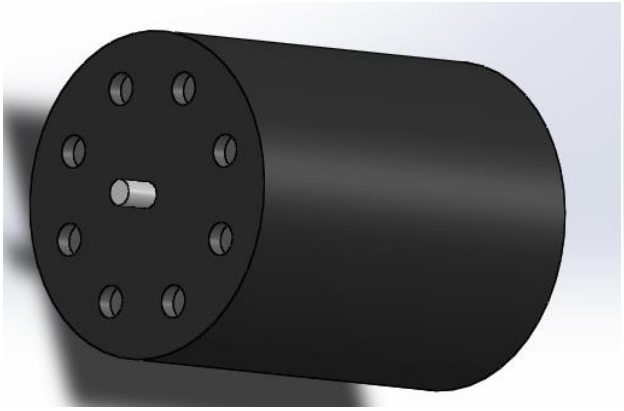


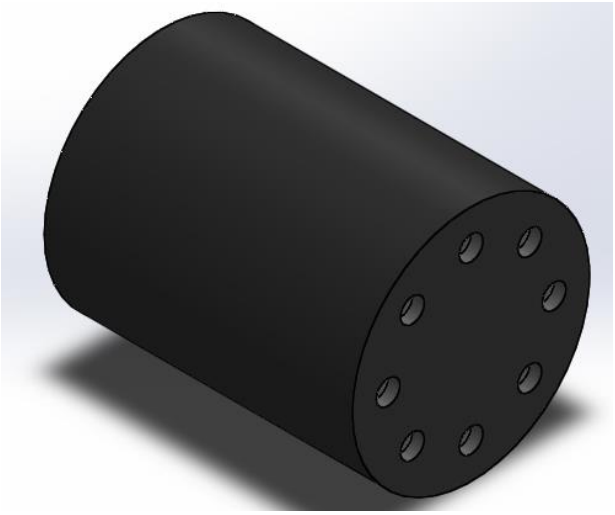
Steering shaft:



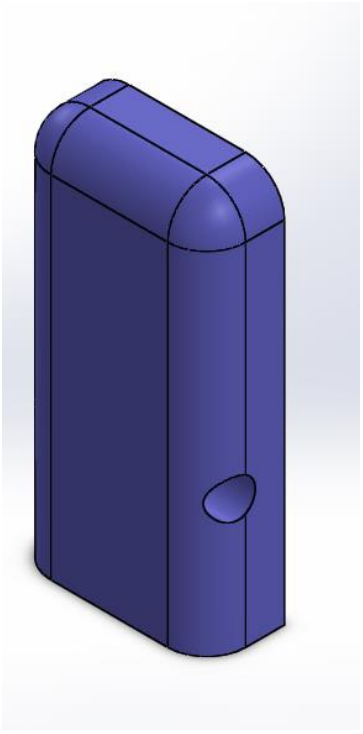


Base motor for motion:

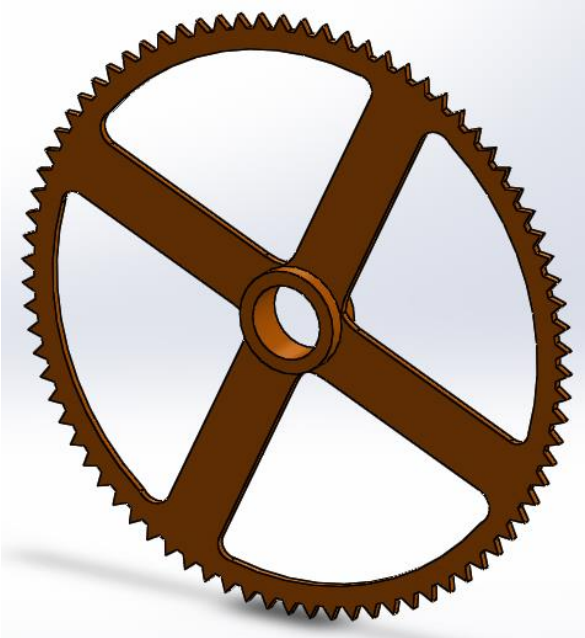




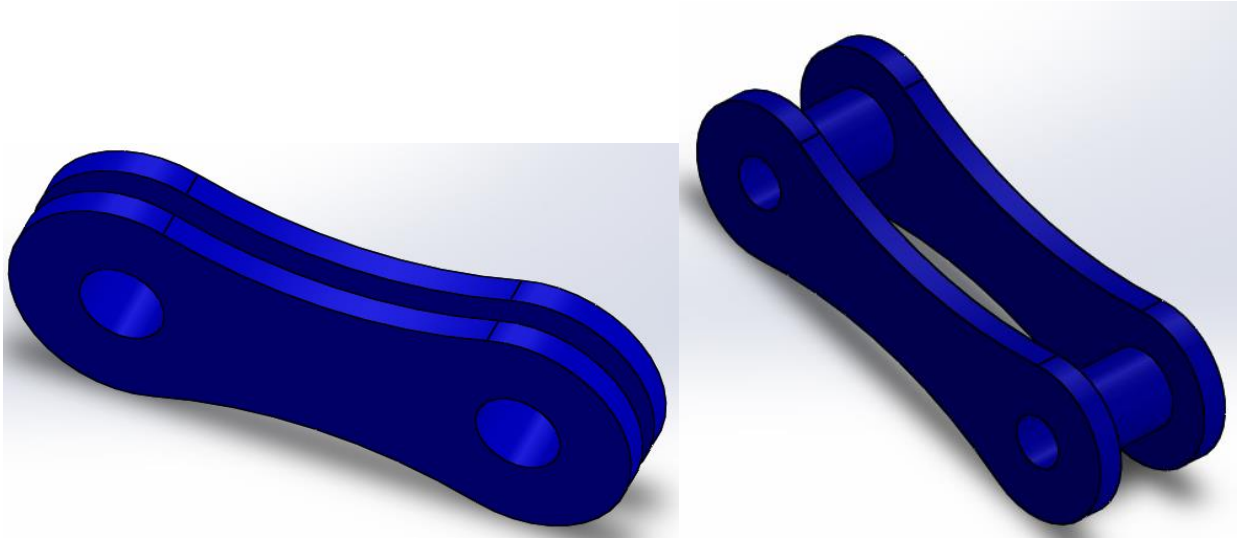
Linear actuator:



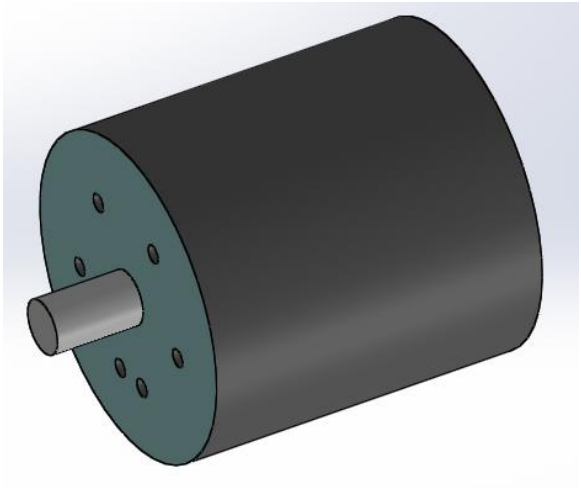
Chain disc:



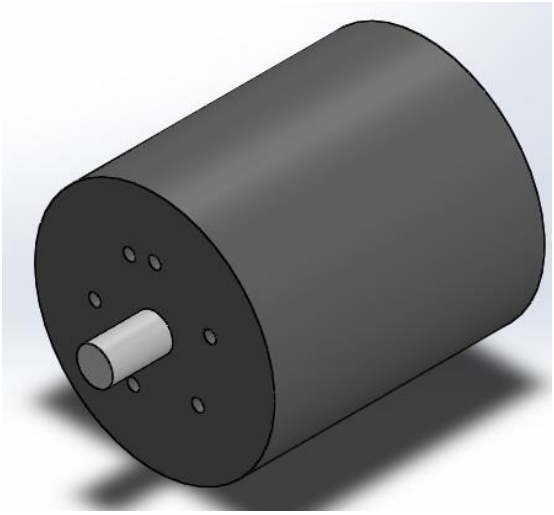
Chain:



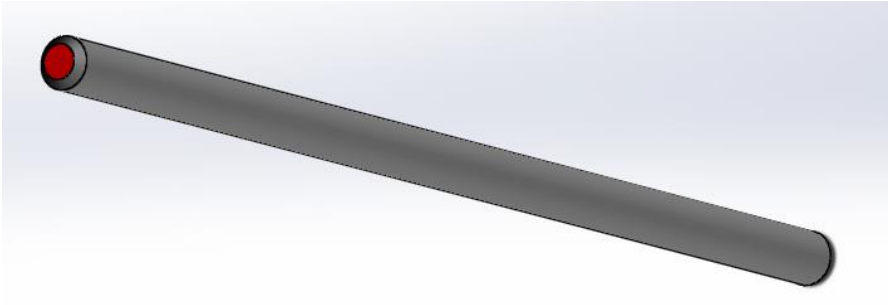
Encoder_1:

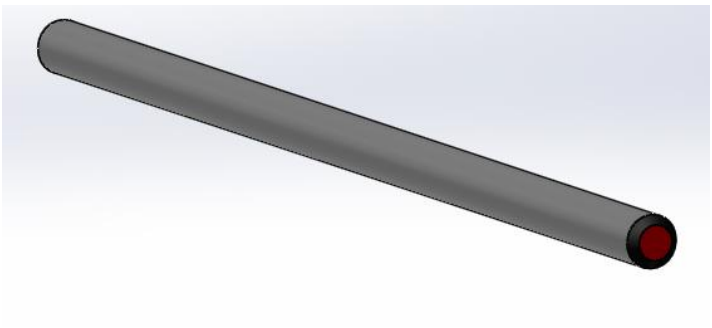


Encoder_2:

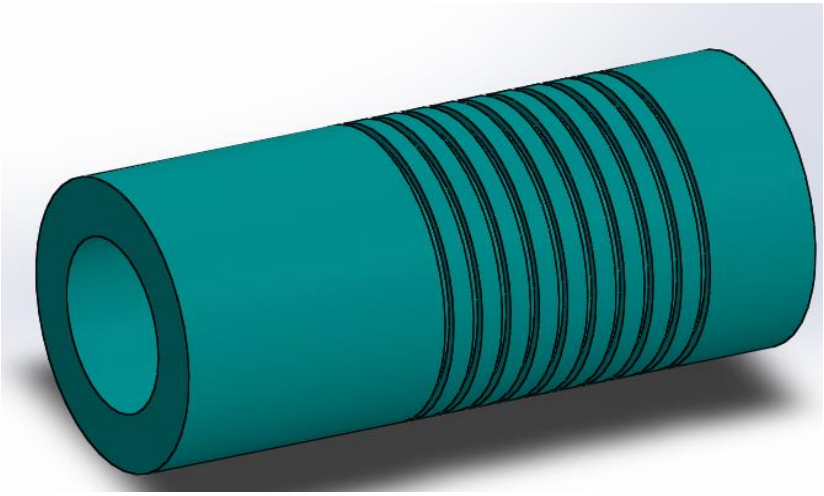


Shaft for back wheel:

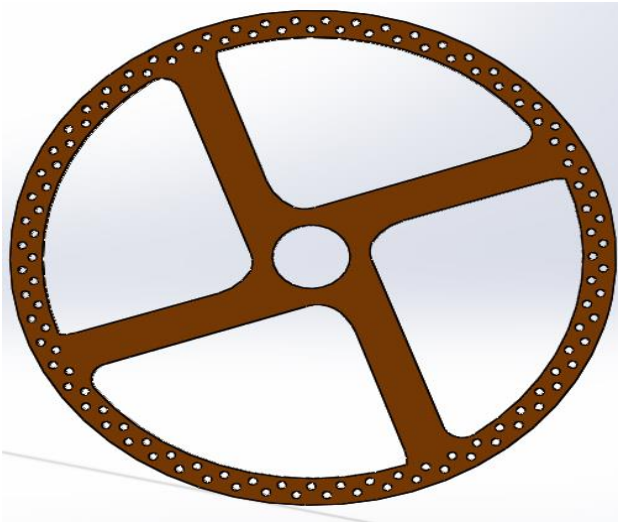




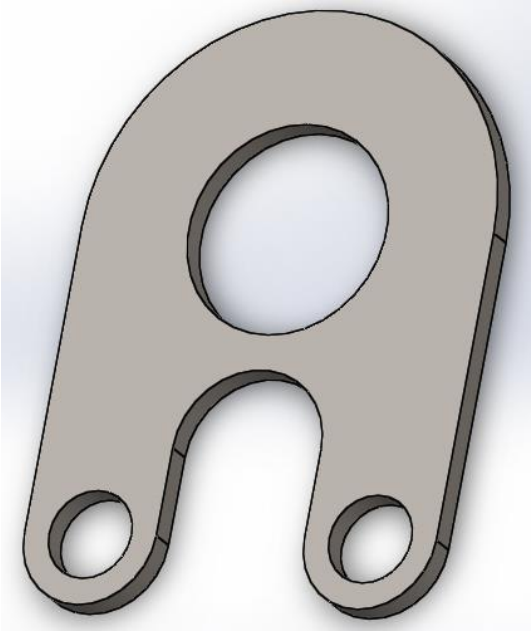
Flexible coupler:



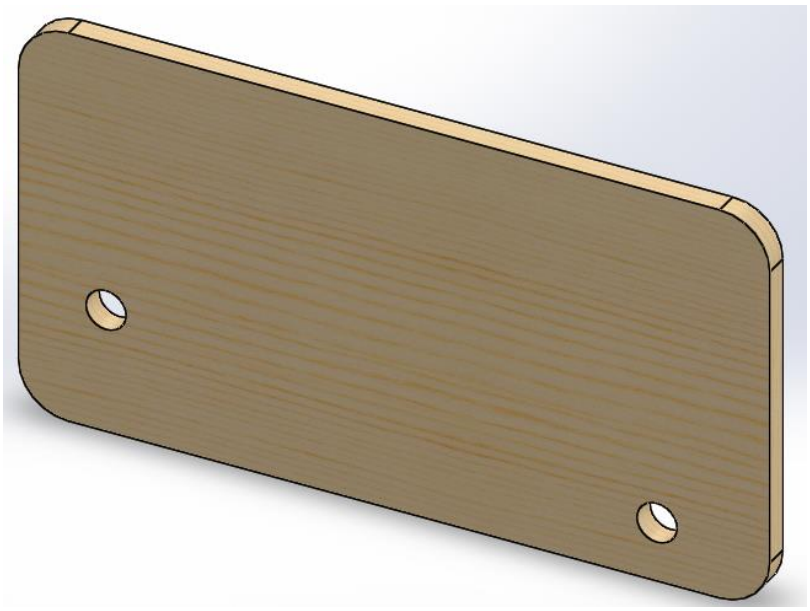
Break disc:



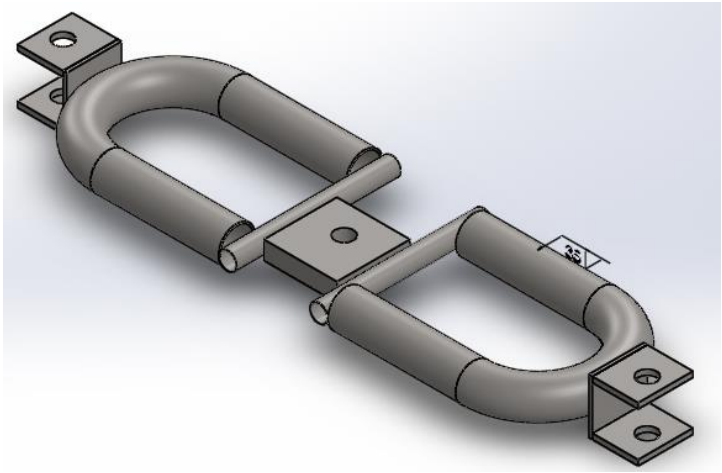
Base for steering shaft:



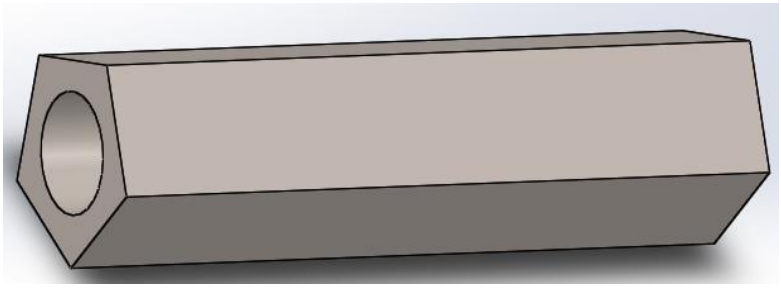
Encoder base:

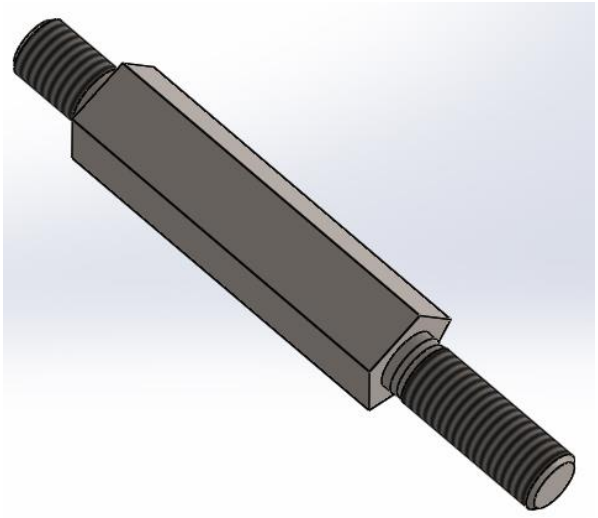


Wheel:

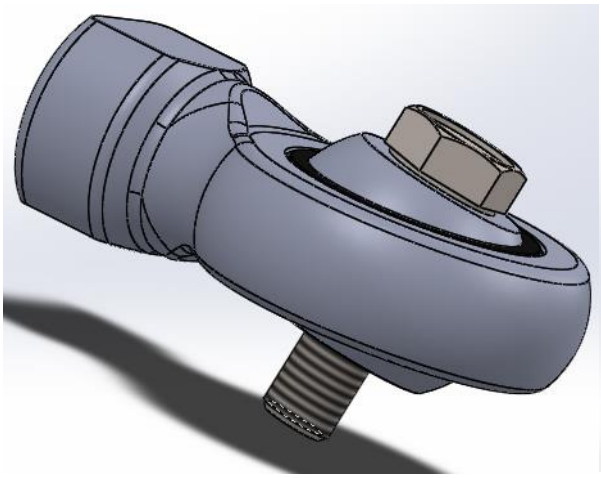


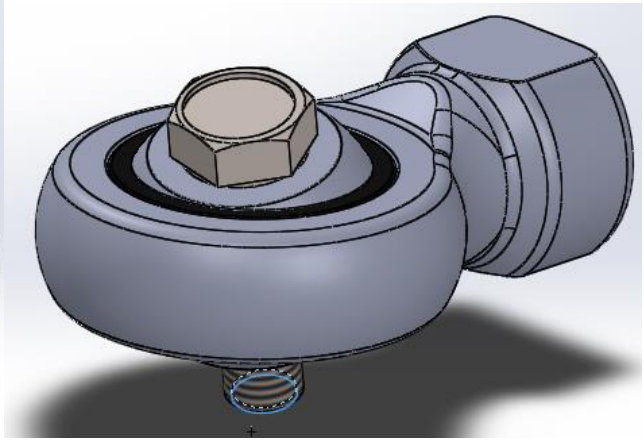
Rod:

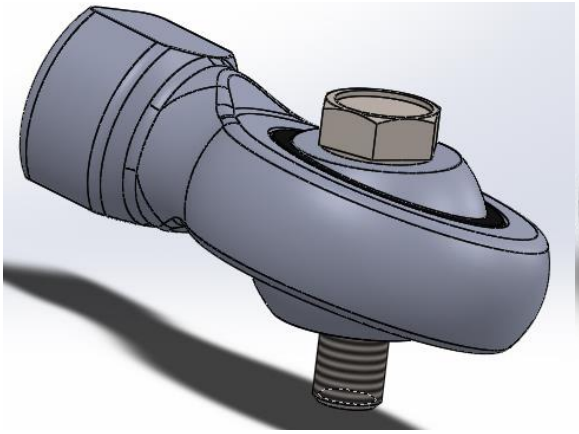


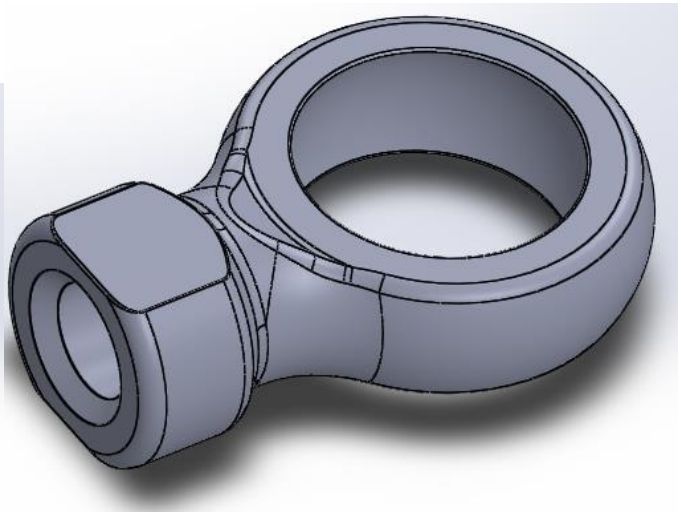


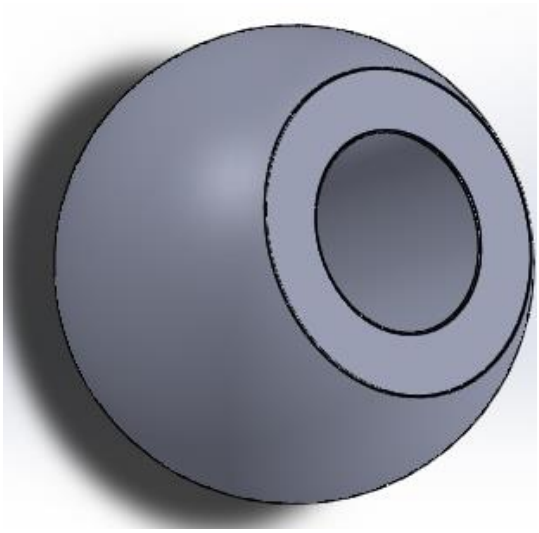
Ball joint:

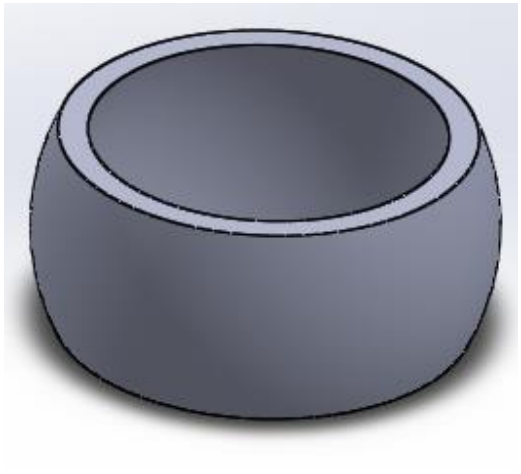




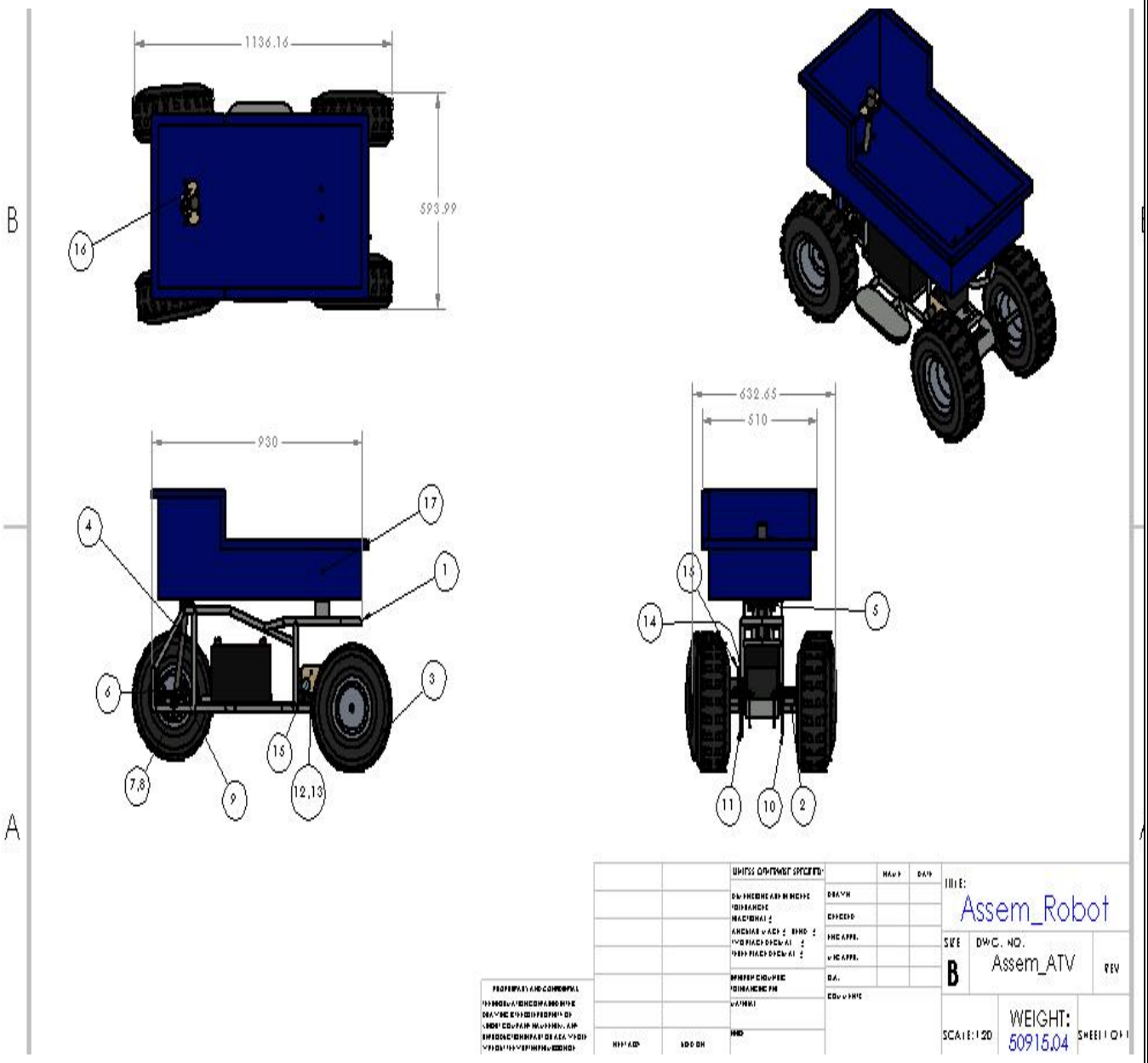




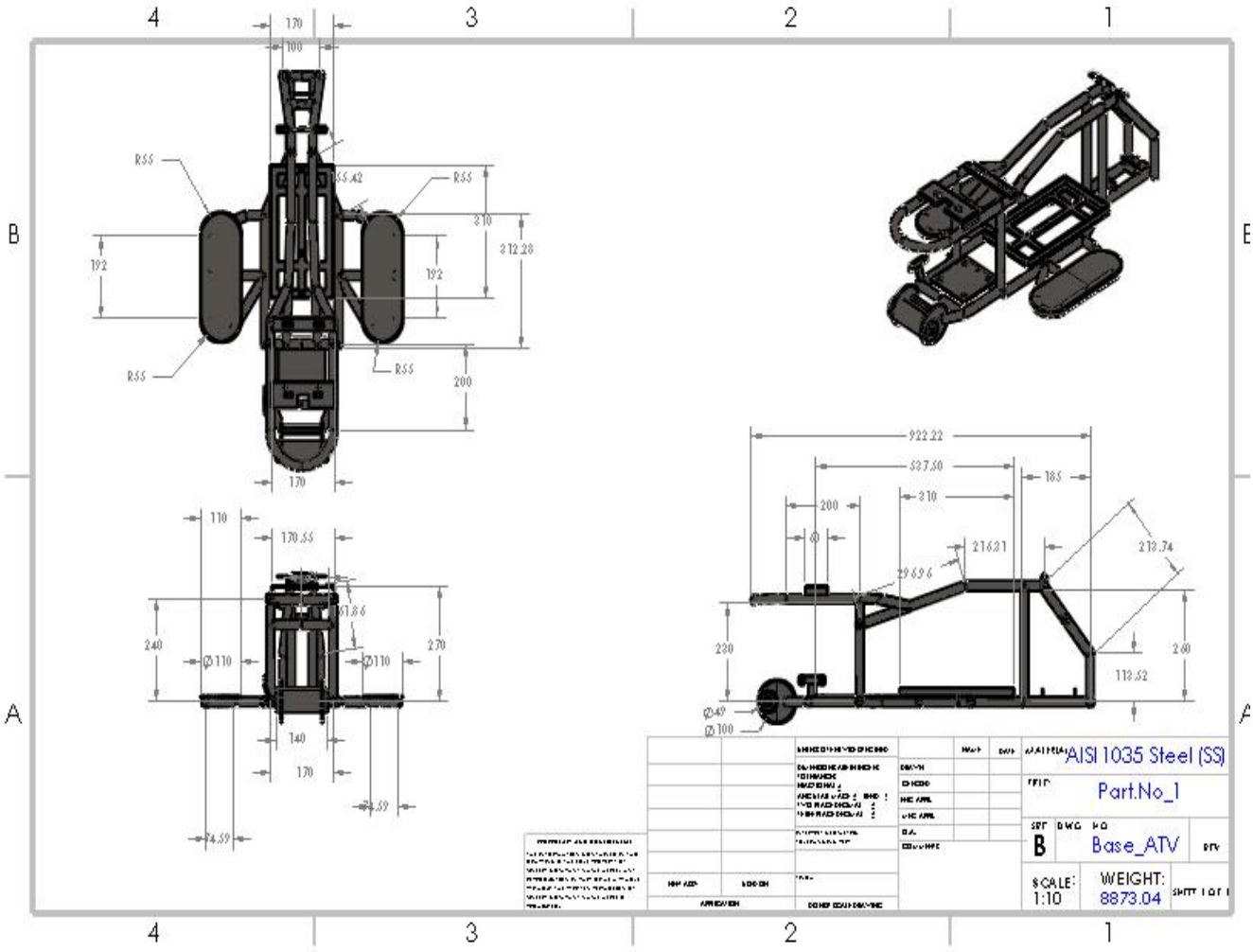




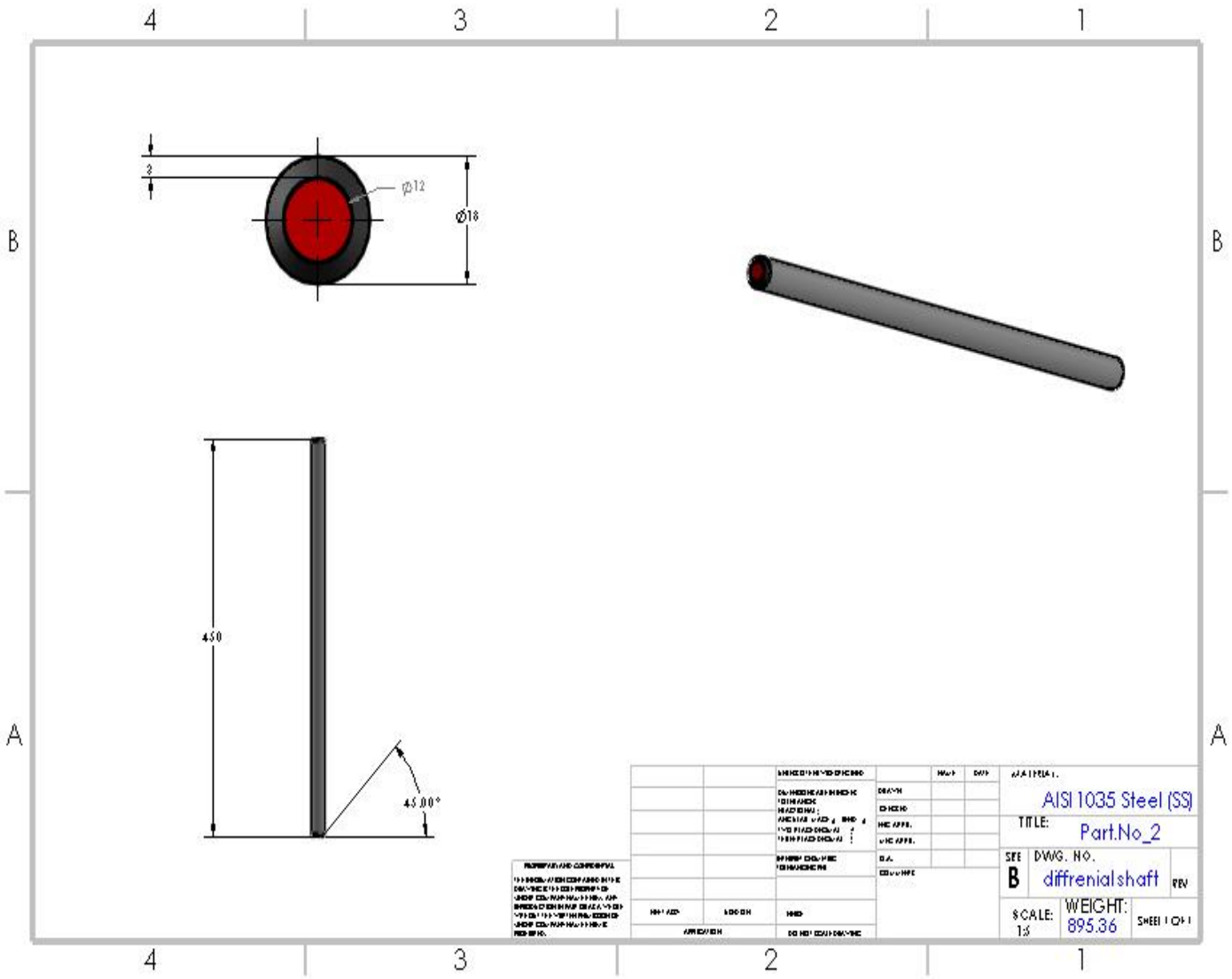
Final design drawing:



1) Base of robot:

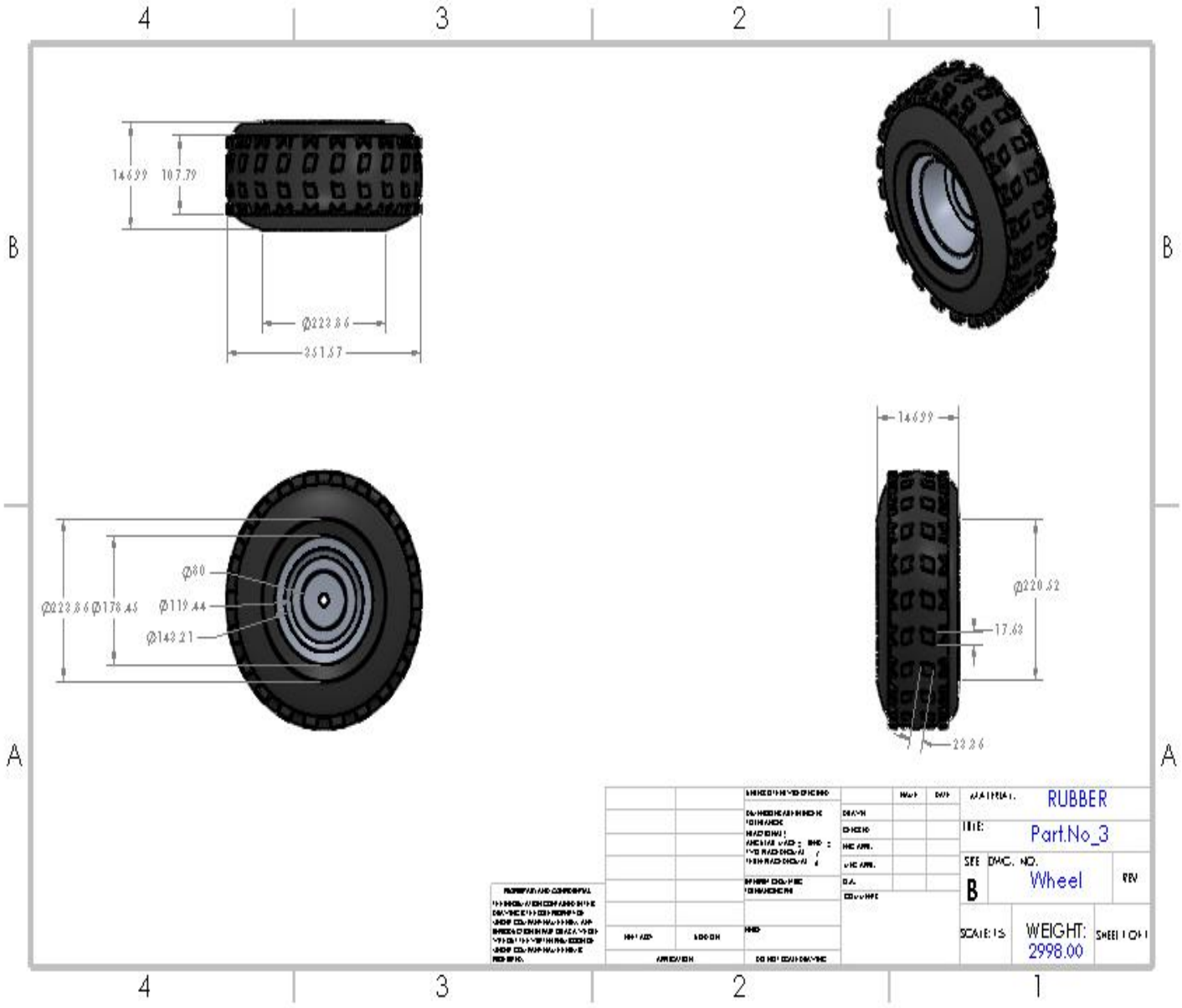


2) Shaft for back wheel :

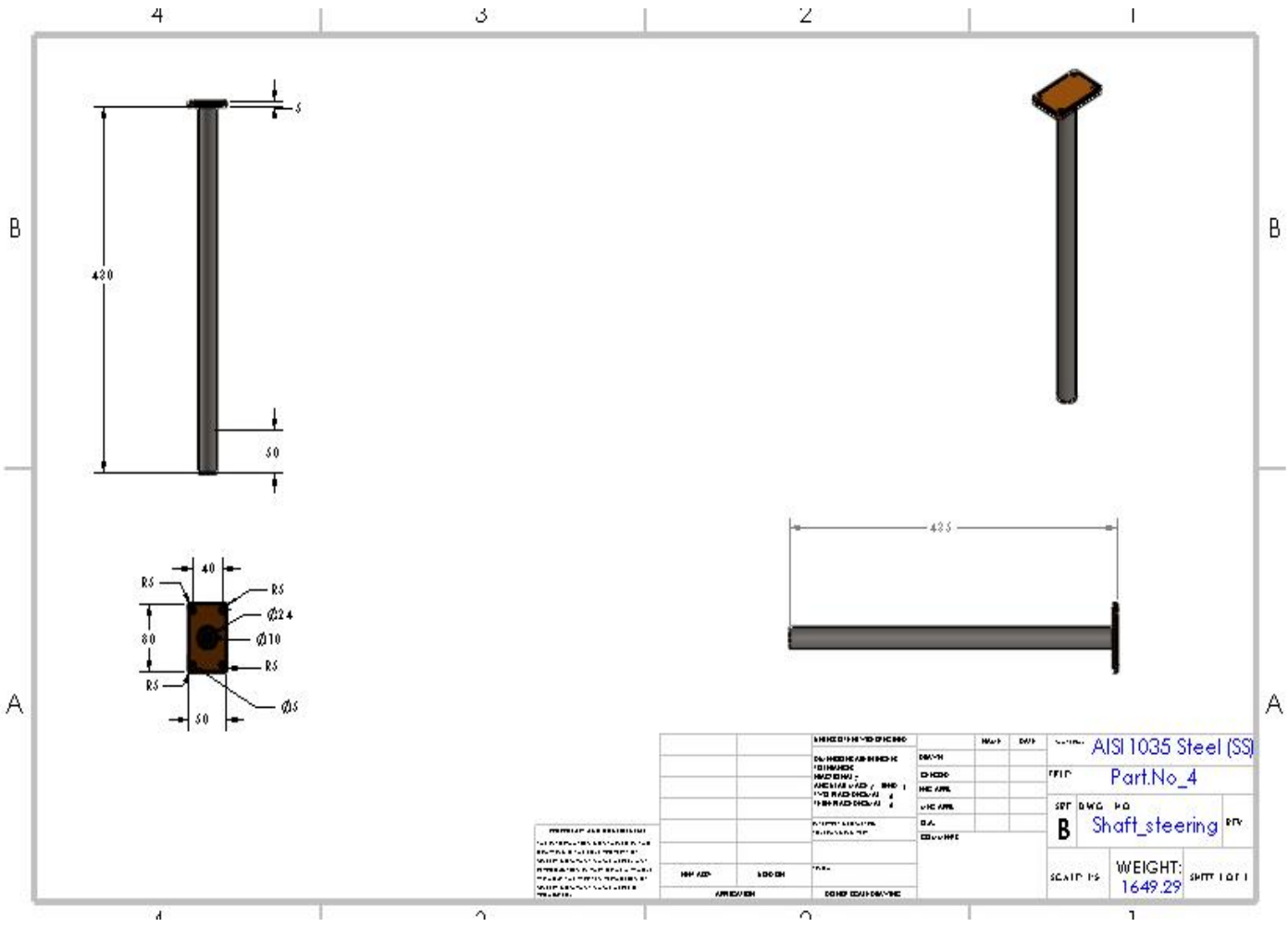


3) Wheels:

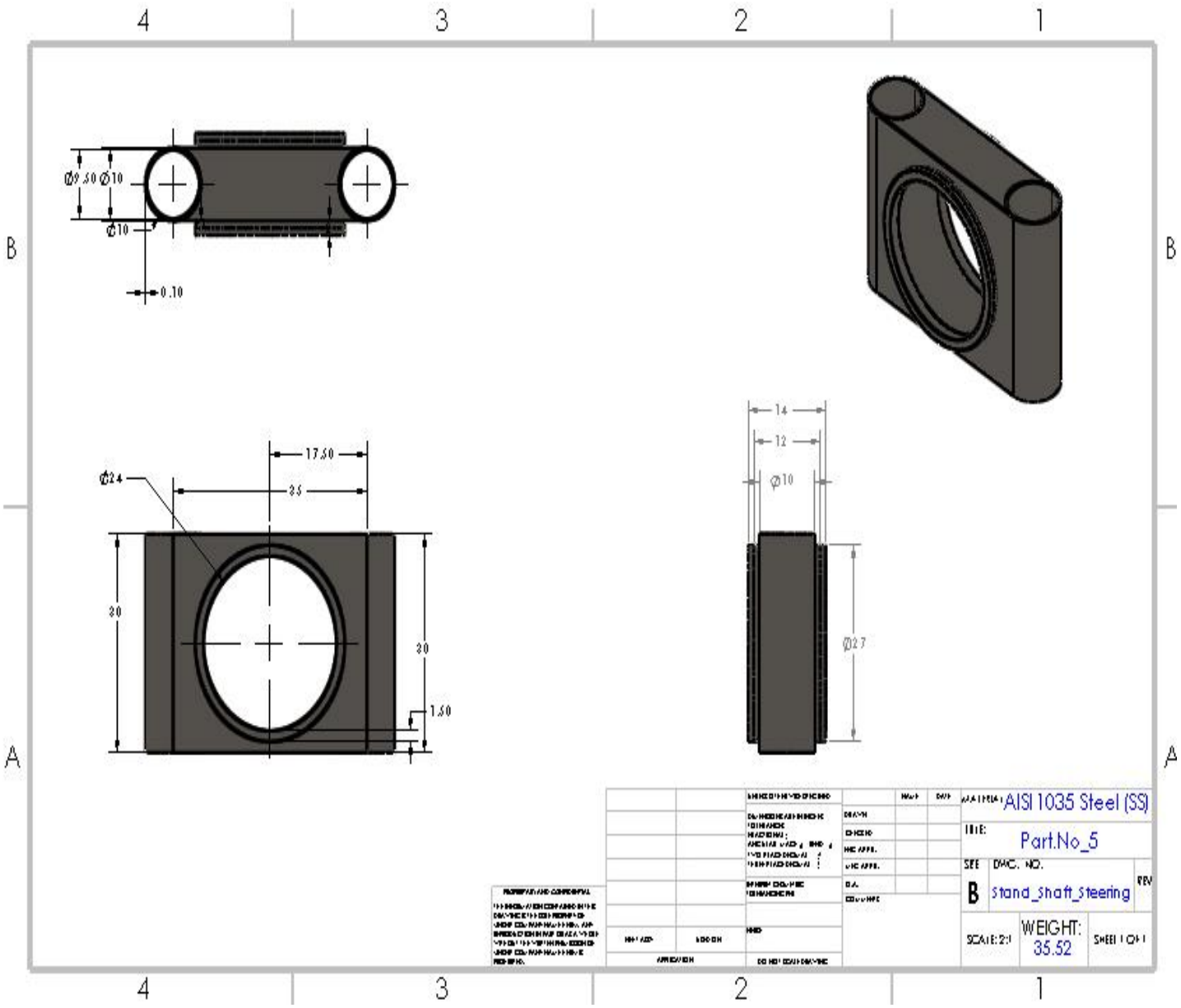
`


4
3
2
1
B
A
14.99
107.79
Ø223.86
251.57
Ø80
Ø119.44
Ø143.21
Ø223.86
Ø178.45
14.99
Ø220.52
17.63
23.26
RUBBER
Part.No_3
Wheel
B
WEIGHT:
2998.00

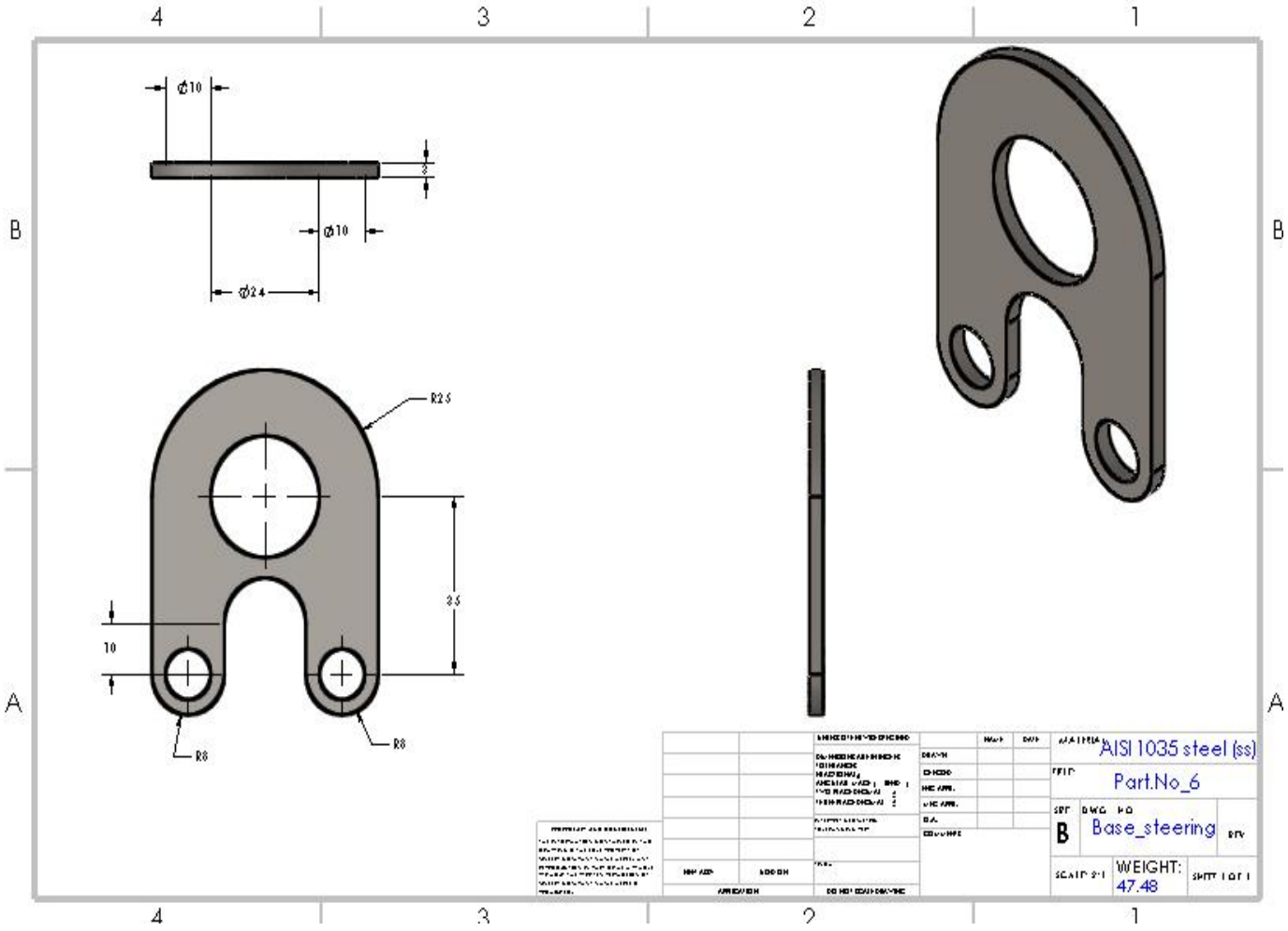
4) Shaft steering



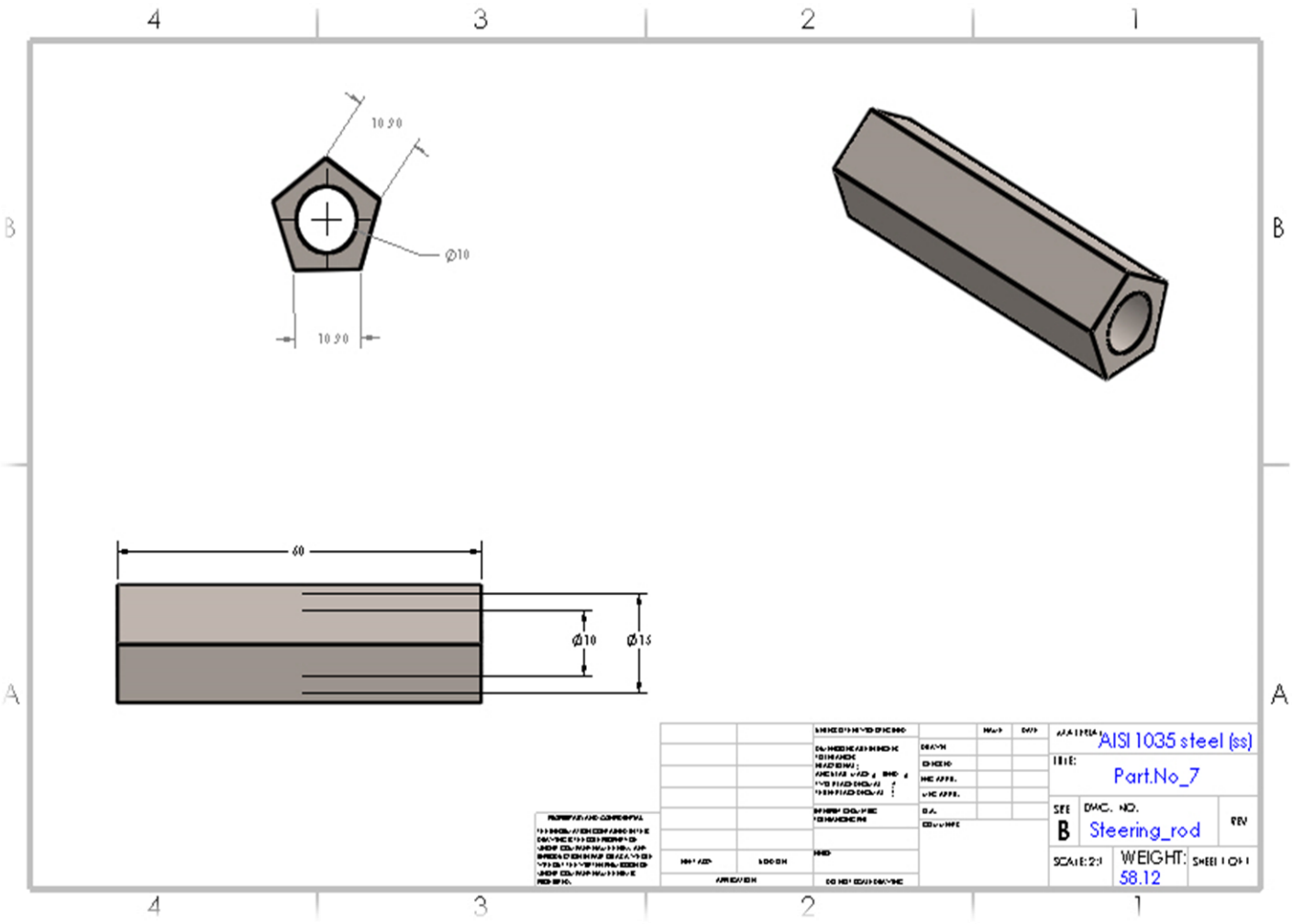
5) Stand shaft steering



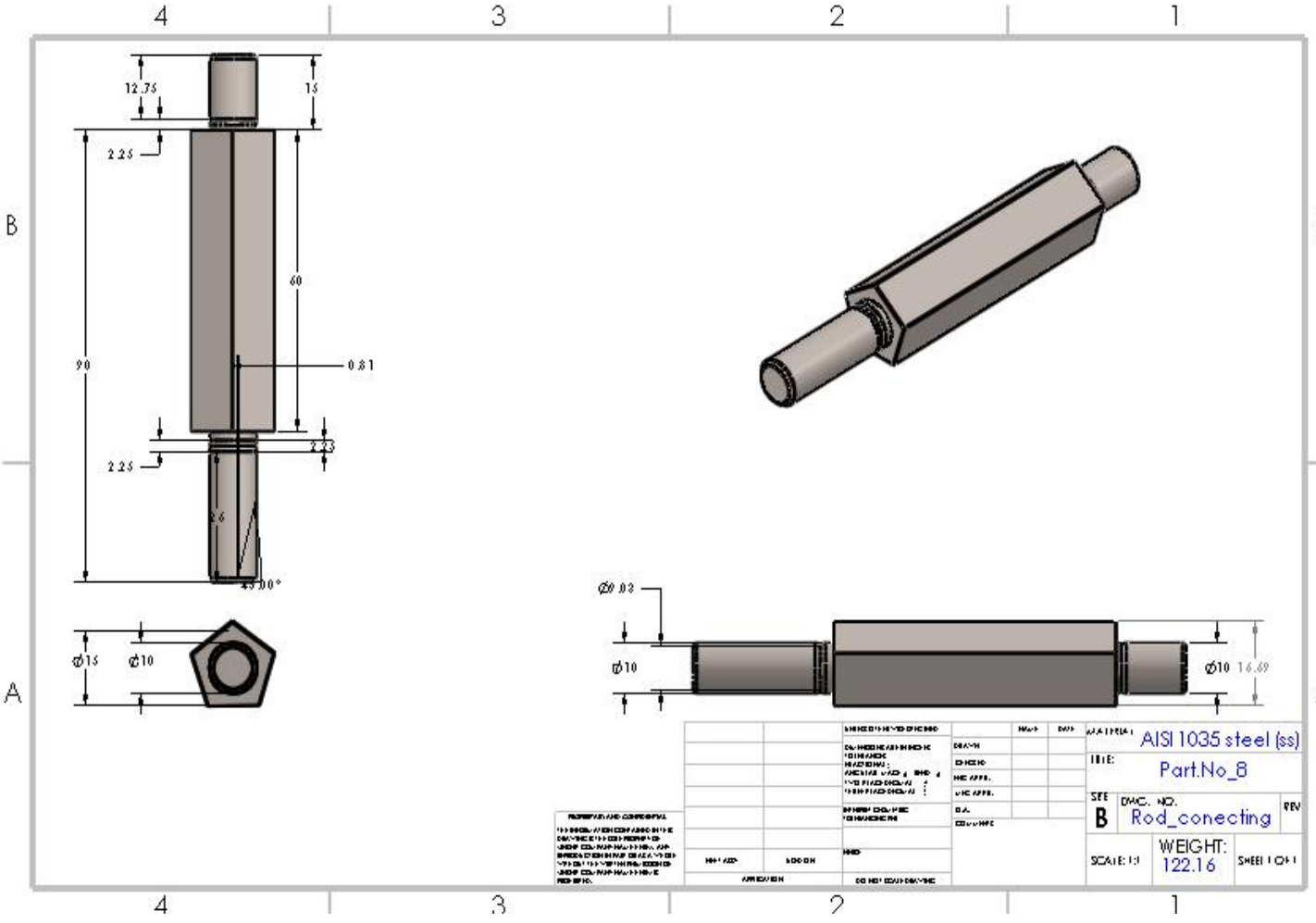
6) Base steering



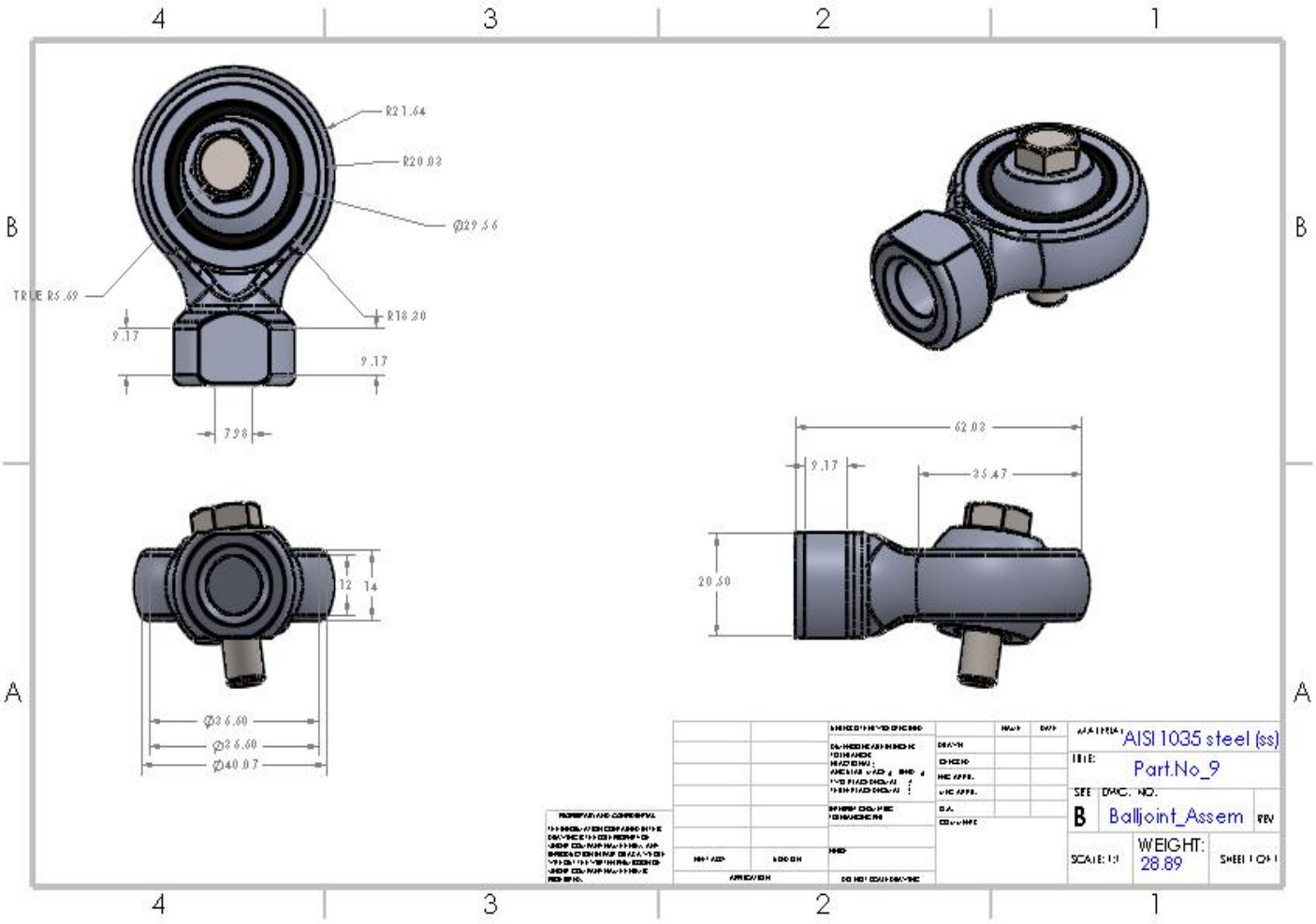
7) Steering rod



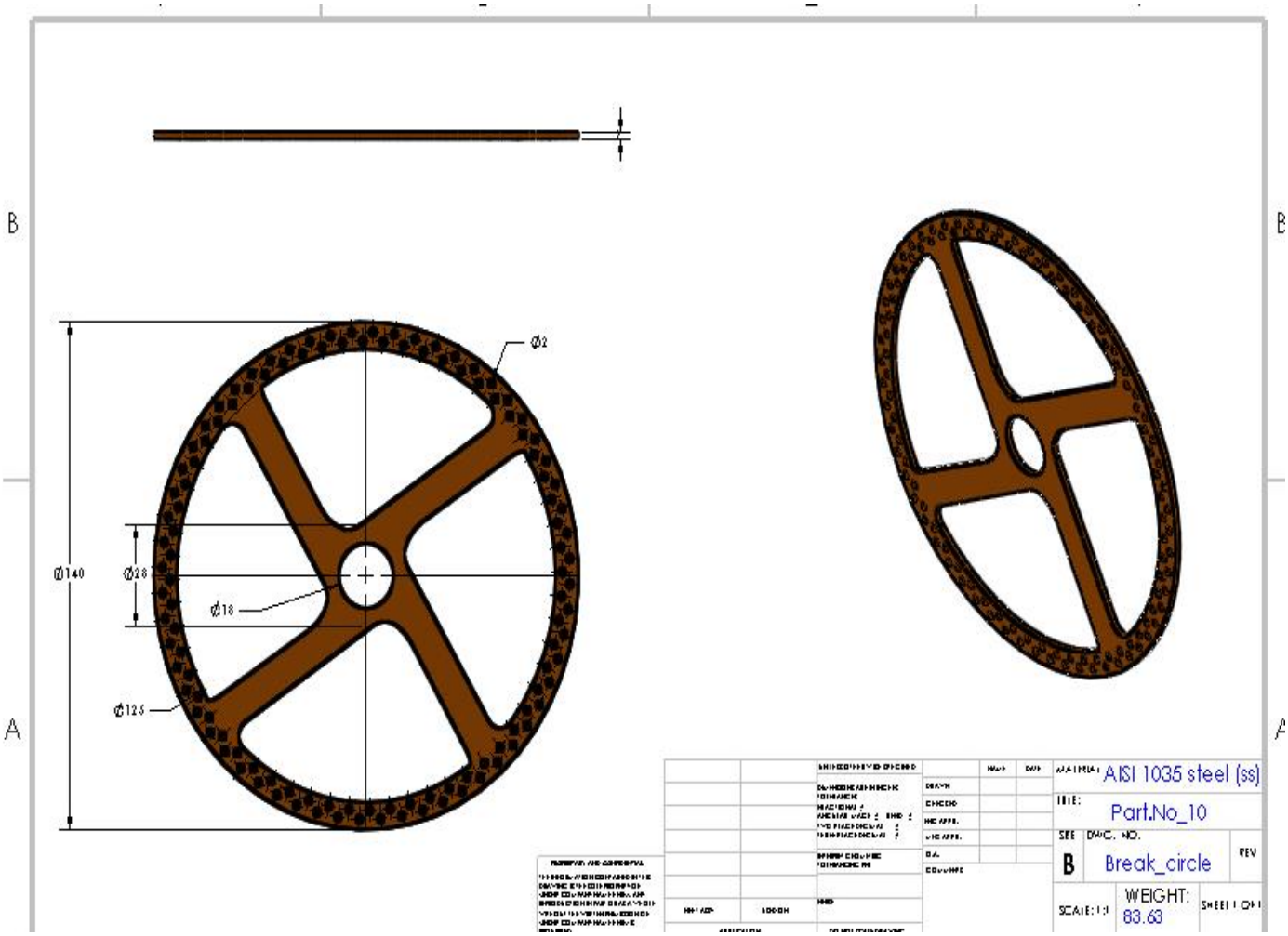
8) Rod connecting



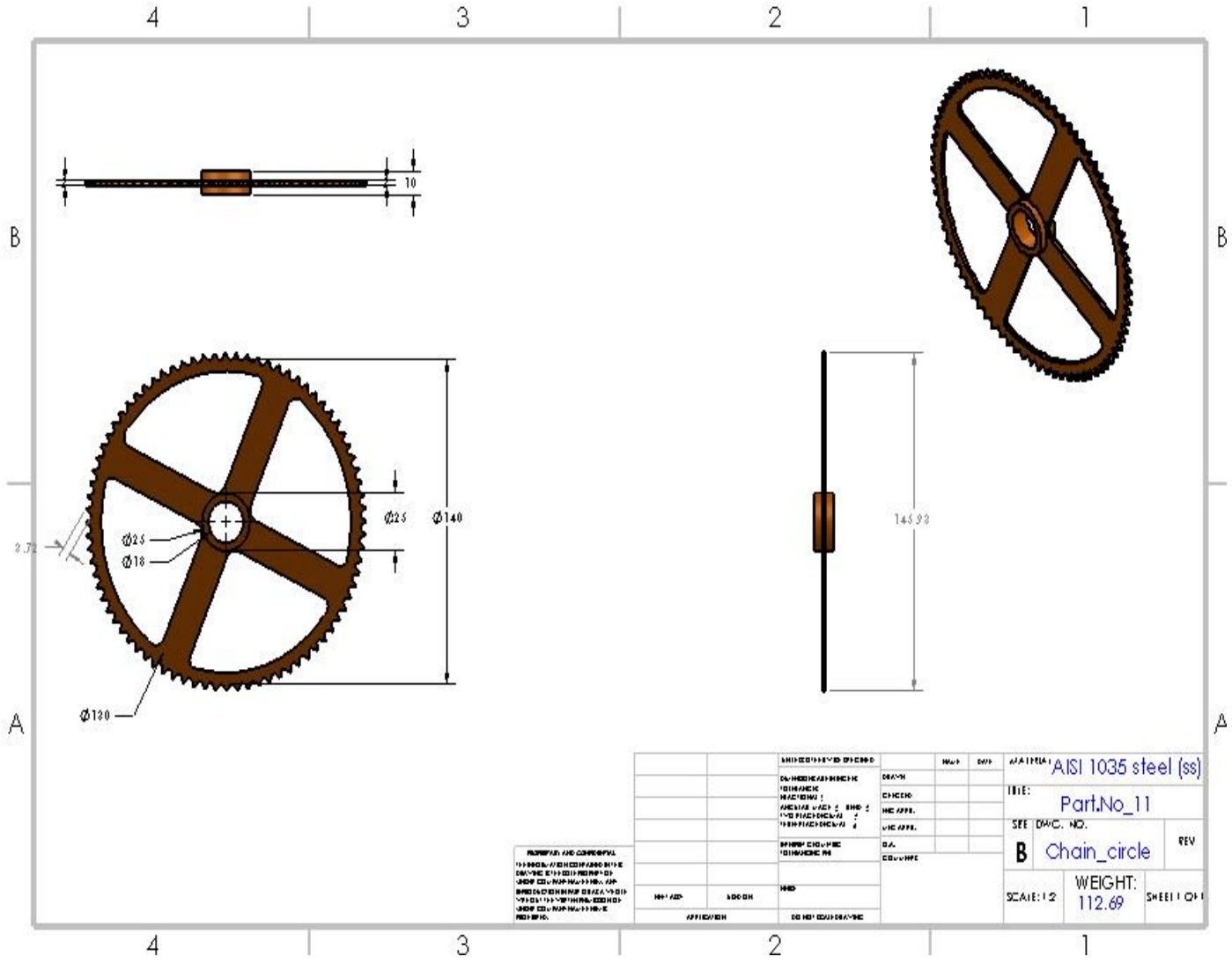
9) Balljoint


R21.64
R20.03
Ø29.56
TRUE R5.69
R18.20
9.17
9.17
7.98
62.03
9.17
35.47
20.50
12
14
Ø36.60
Ø36.60
Ø40.07
AISI 1035 steel (ss)
Part.No_9
B
Balljoint_Assem
WEIGHT:
28.89

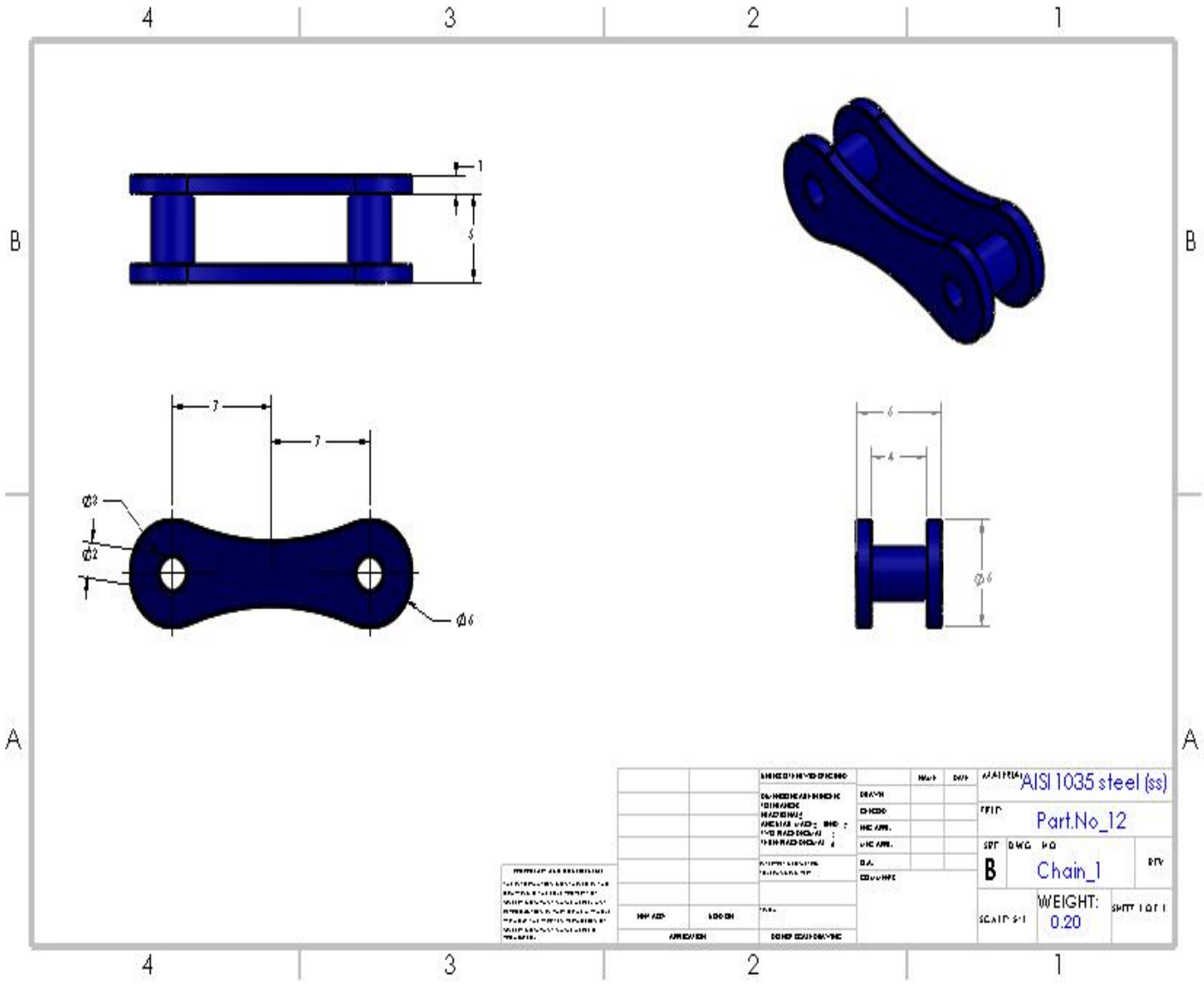
10) Break circle



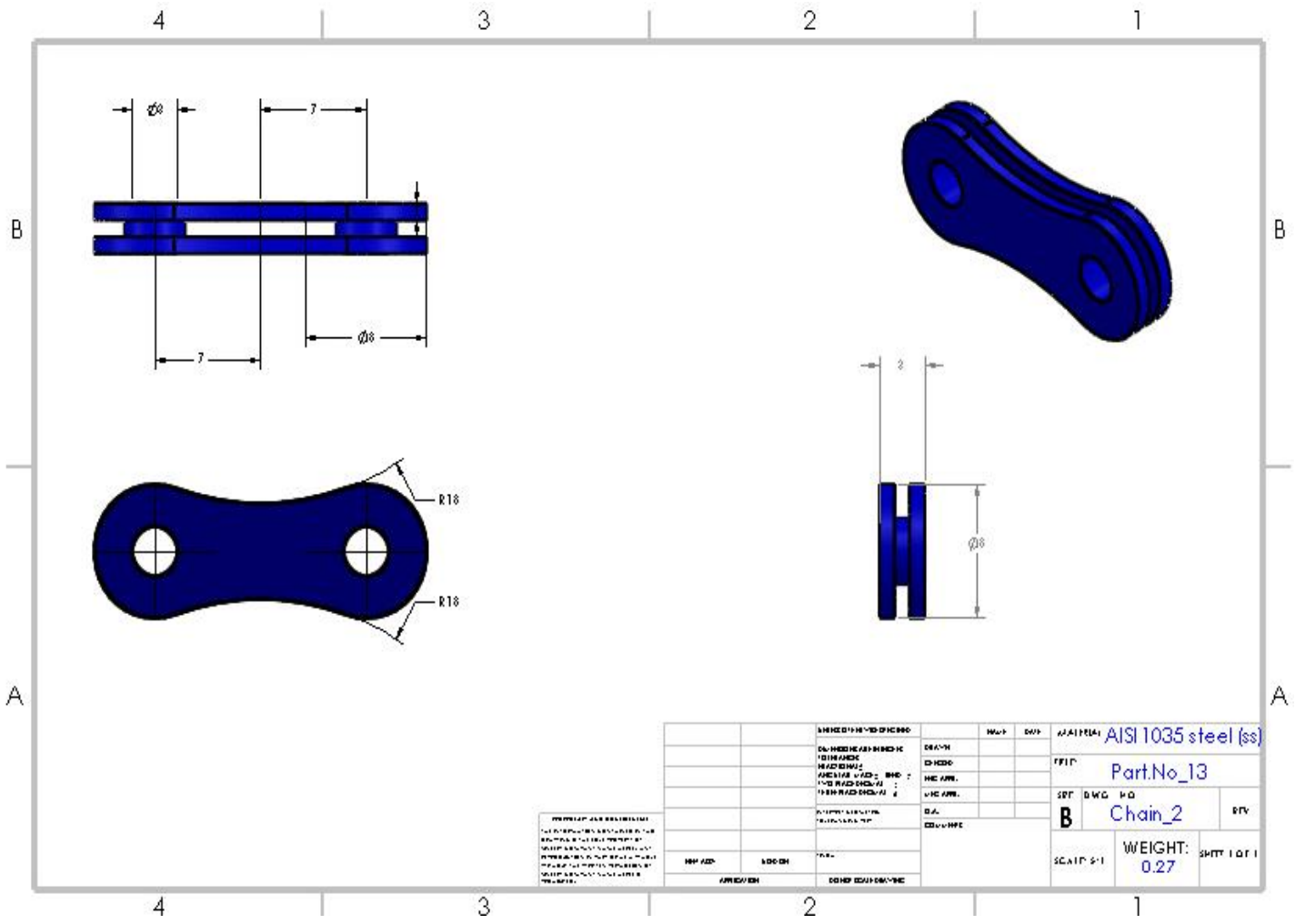
11) chain circle



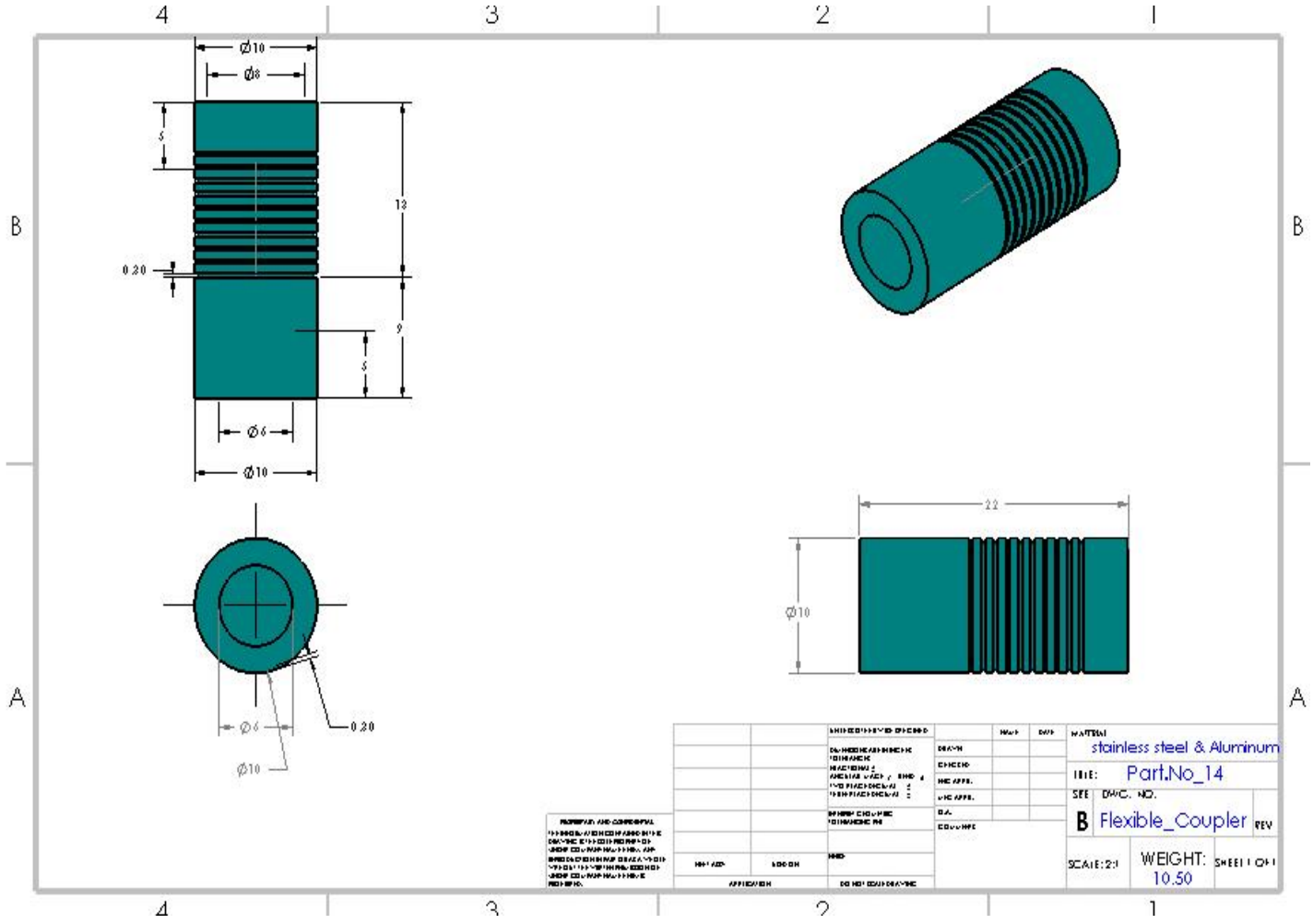
12) chain_1



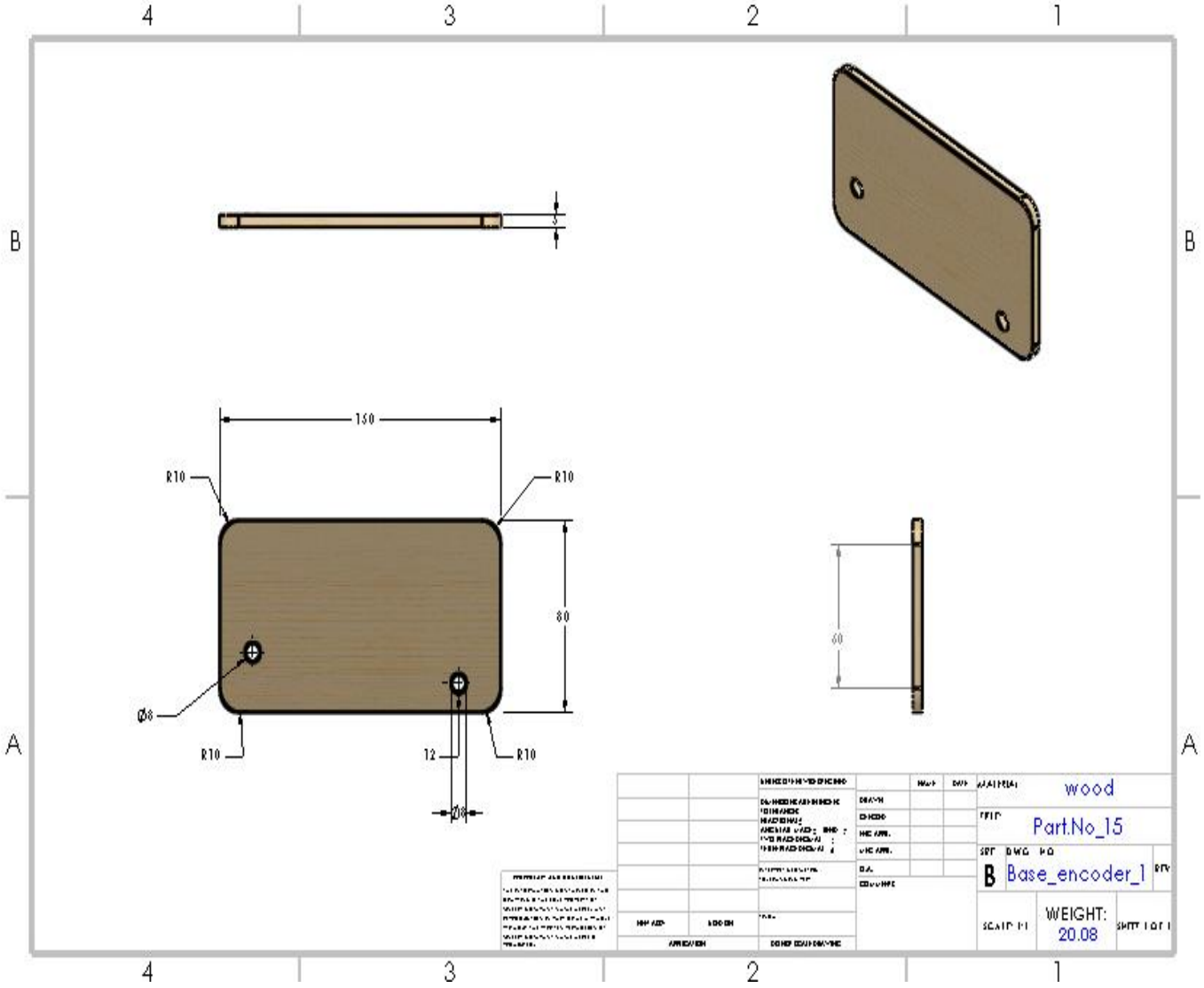
13) Chain_2


4
3
2
1
B
A
Ø8
7
7
Ø8
R18
R18
2
Ø8
AISI 1035 steel (ss)
Part.No_13
B
Chain_2
WEIGHT:
0.27

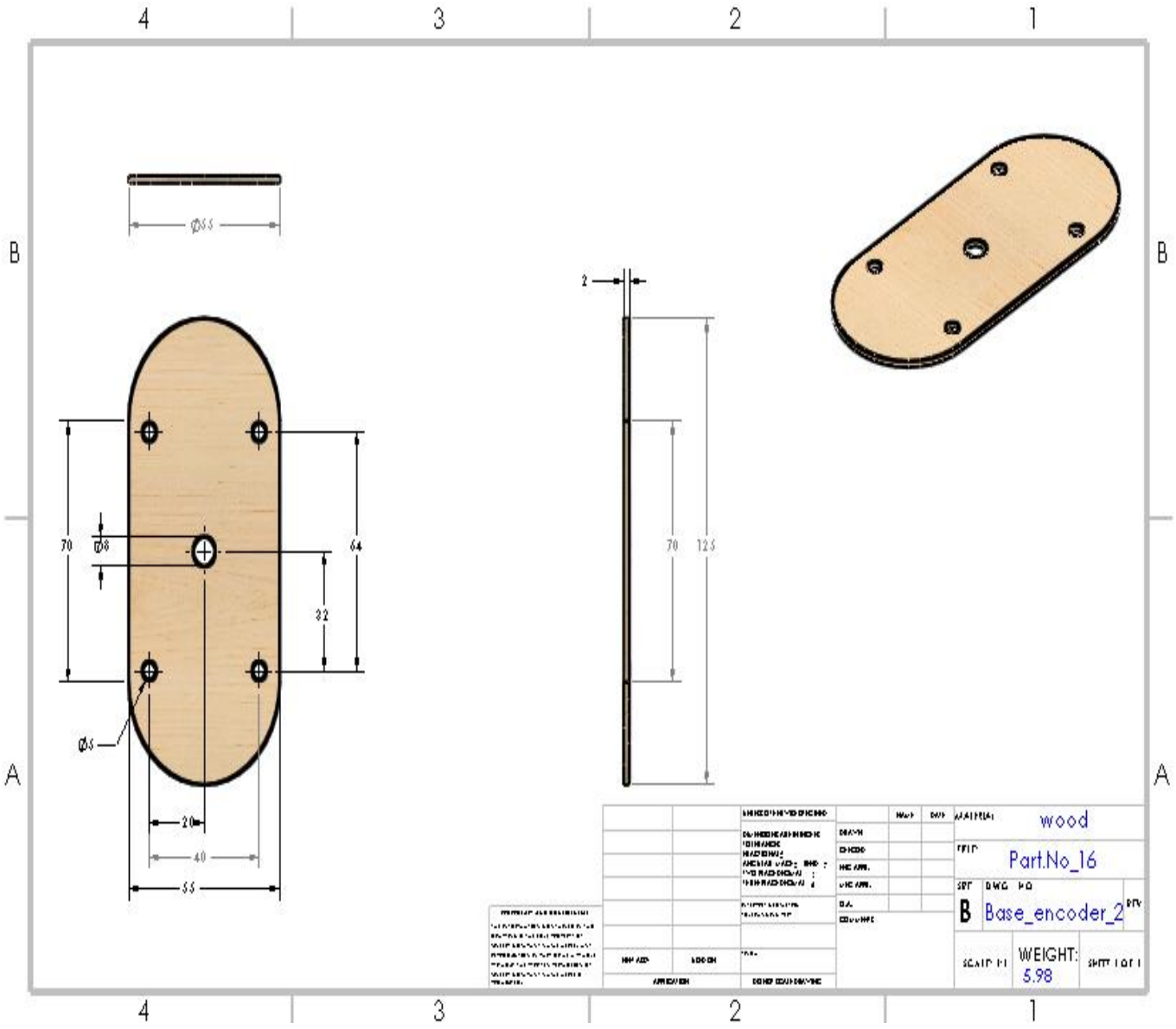
14) Flexible coupler



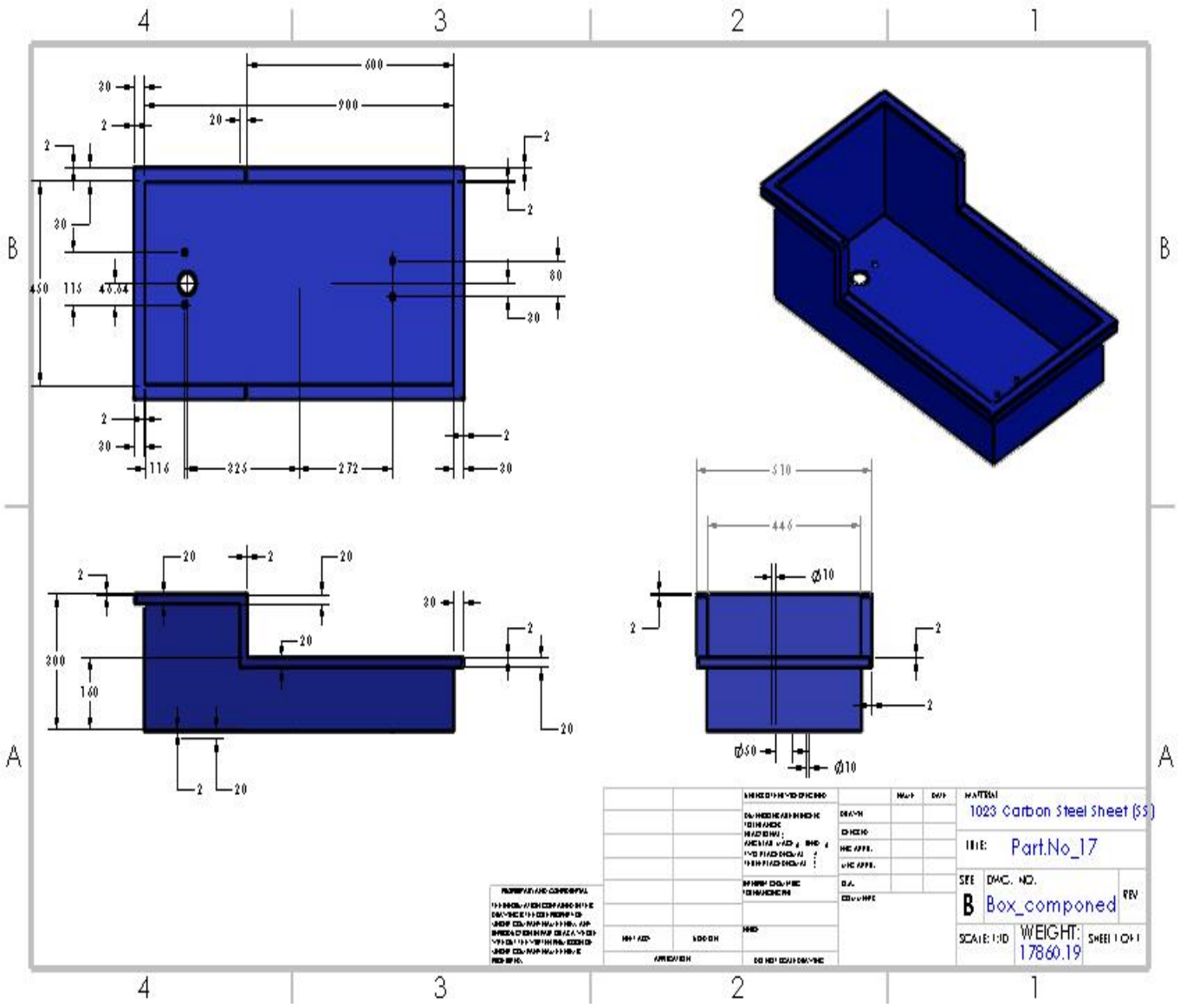
15) Base encoder_1



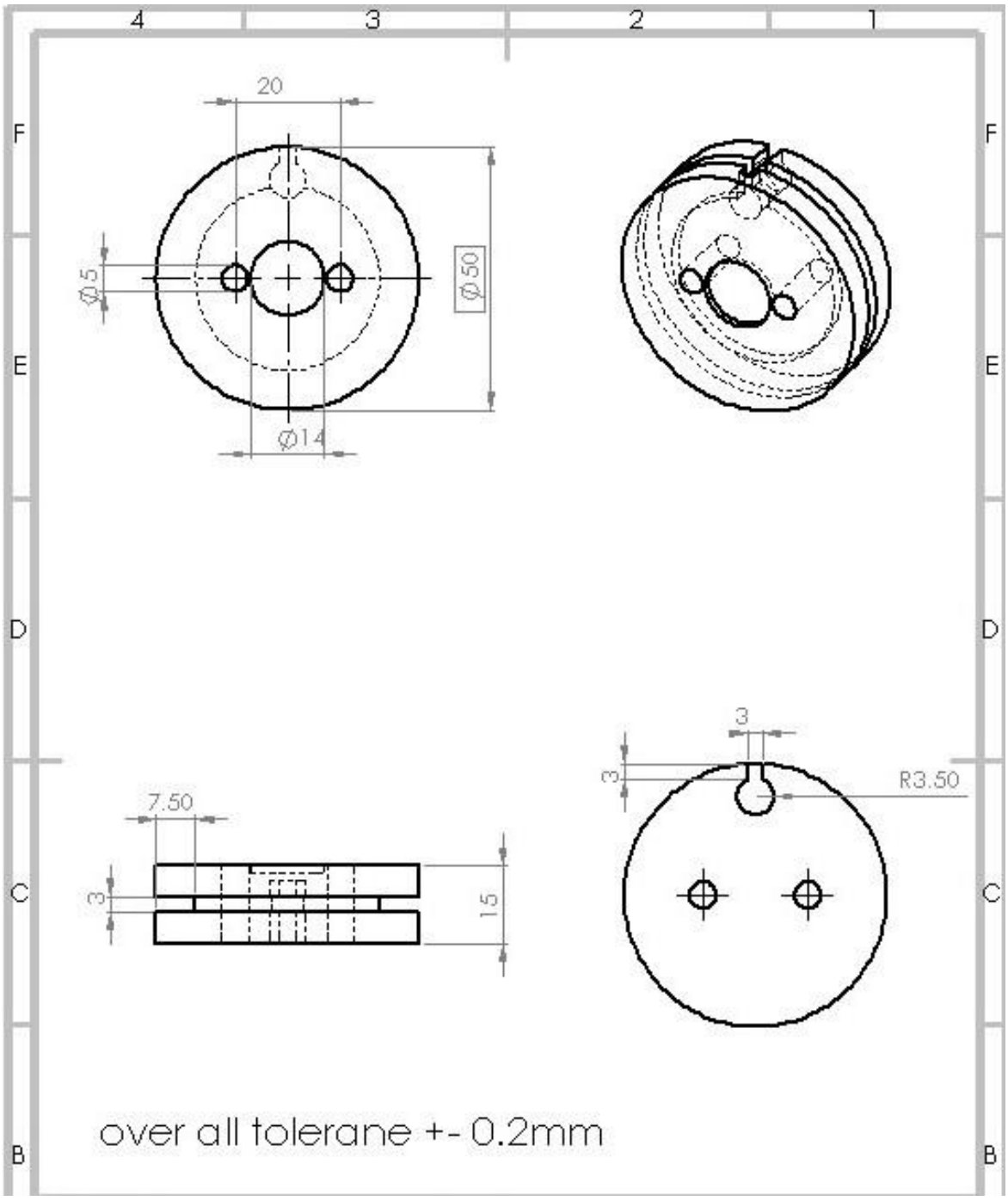
16) Base encoder_2



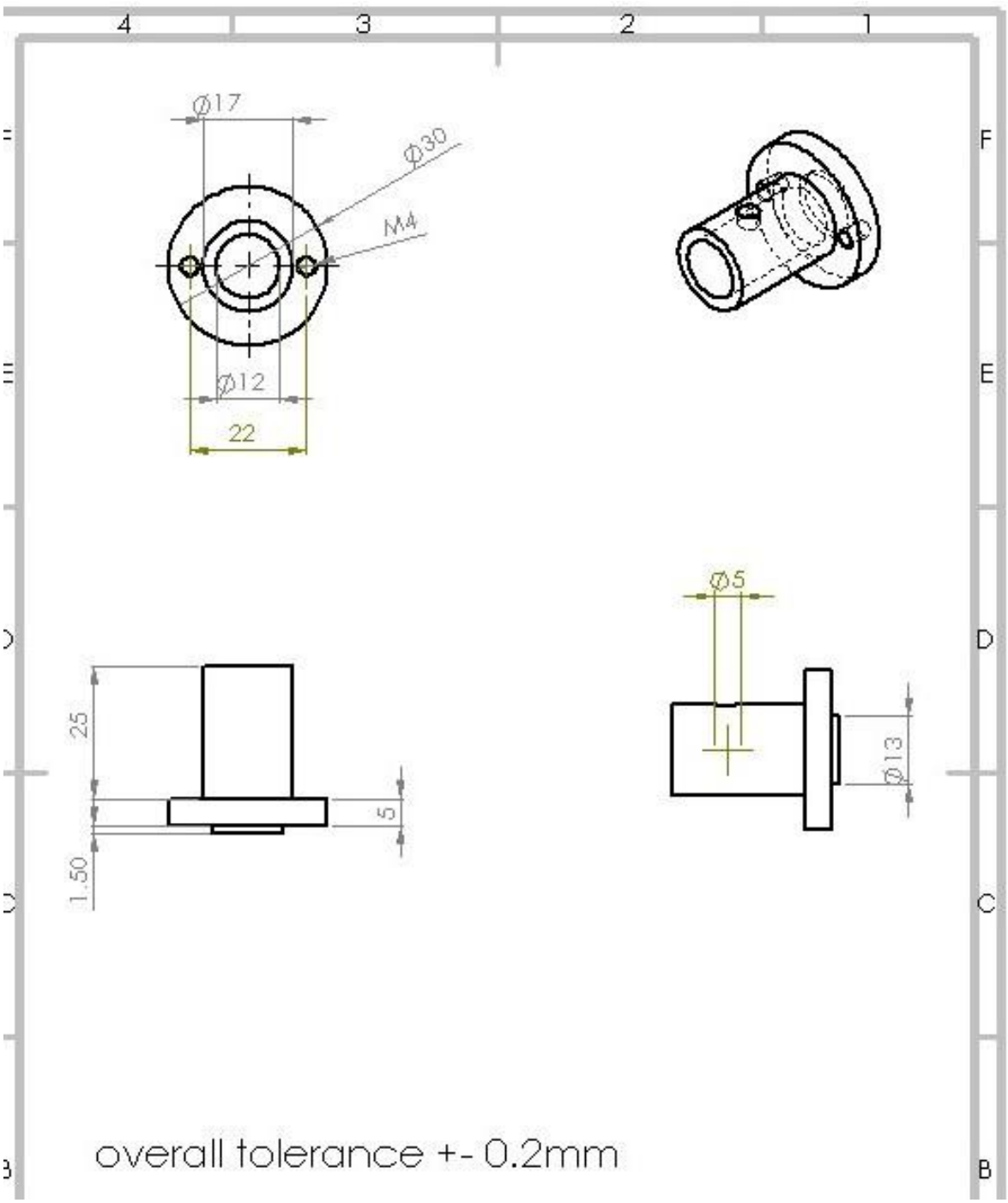
17) Box for compounds:



18) Roller


4
3
2
1
F
E
D
C
B
20
Ø5
Ø50
Ø14
7.50
3
15
3
3
R3.50
over all tolerane +- 0.2mm

19) Roller Coupler



References

[1] MCCA global tec fourm, "Autonomous Vehicles;Navigating the legal and regulatory issues".

[2] i. t. fourm, "automated and autonomous driving Regulation under uncertainty".

[3] semanticscholar, "semanticscholar," [Online]. Available: semanticscholar.com .

[4] techtarget, "internetofthingsagenda," [Online]. Available: internetofthingsagenda.techtarget.com.

[5] elprocus, "elprocus," [Online]. Available: elprocus.com.

[6] microchip, "microchip," [Online]. Available: microchip.com.

[7] dynapar, "dynapar," [Online]. Available: dynapar.com.

[8] ni, "ni," [Online]. Available: ni.com.

[9] microchipdeveloper, "microchipdeveloper," [Online]. Available: https://microchipdeveloper.com/8avr:ioports.

[10] components101, "components101," [Online]. Available: https://components101.com/microcontrollers/atmega32-8-bit-avr-microcontroller.

[11] ece.utexas, "ece.utexas," [Online]. Available: http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C12_Interrupts.htm.

[12] circuitdigest, "circuitdigest," [Online]. Available: https://circuitdigest.com/tutorial/what-is-pwm-pulse-width-modulation.

[13] circuitbasics, "circuitbasics," [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/.

[14] sciencedirect, "sciencedirect," [Online]. Available: https://www.sciencedirect.com/topics/engineering/incremental-encoder.

[15] alldatasheet, "alldatasheet," [Online]. Available: https://html.alldatasheet.com/html-pdf/77378/ATMEL/ATMEGA32/259/2/ATMEGA32.html.

[16] I. school.

[17] astromachineworks, "astromachineworks," [Online]. Available: astromachineworks.com.

[18] insights.globalspec, "insights.globalspec," [Online]. Available: insights.globalspec.com.

[19] auto.howstuffworks, "auto.howstuffworks," [Online]. Available: auto.howstuffworks.com.

[20] wcroberts, "wcroberts," [Online]. Available: wcroberts.com.

[21] dgrelectriccylinder, "dgrelectriccylinder," [Online]. Available: dgrelectriccylinder.com.

[22] "The Ackermann Principle as Applied to Steering".
[23] Esselink, B. (1998). A practical guide to software localization: for translators, engineers and project managers. Amsterdam: John Benjamins Pub.
[24] Naidu, P. S. (2018). Distributed sensor arrays: localization. Boca Raton: CRC Press, Taylor & Francis Group
[25] Teskey, W. (2008). Precision six degree-of-freedom motion tracking using inertial sensors. Ottawa: Library and Archives Canada = Bibliothèque et Archives Canada
[26] Teskey, W. (2008). Precision six degree-of-freedom motion tracking using inertial sensors. Ottawa: Library and Archives Canada = Bibliothèque et Archives Canada
[27] Arfianto, A. Z., Rahmat, M. B., Setiyoko, A. S., Handoko, C. R., Hasin, M. K., Utari, D. A., … Aminudin, A. (2018). Perangkat Informasi Dini Batas Wilayah Perairan Indonesia Untuk Nelayan Tradisional Berbasis Arduino Dan Modul Gps Neo-6M. Joutica, 3(2), 163. doi: 10.30736/jti.v3i2.229
[28] Kharisma, O. B., Dzikra, A. A., Mustakim, Vebrianto, R., Novita, R., Hasbullah, … Andriani, T. (2019). Development of location tracking system via short message service (SMS) based on GPS unblox neo-6m and sim 800l module. Journal of Physics: Conference Series, 1363, 012002. doi: 10.1088/1742-6596/1363/1/012002
[29] Hager, J. W. (1989). The Universal grids: universal transverse mercator (Utm) and universal polar stereographic (Ups). Fairfax, VA: Defense Mapping Agency, Hydrographic/Topographic Center
[30] The Universal Transverse Mercator (UTM) Grid. (2001). Fact Sheet. doi: 10.3133/fs07701
[31] Groves, P. D. (2013). Principles of Gnss, inertial, and multisensor integrated navigation systems. Boston: Artech House
[32] NEO-6M GPS datasheet
[33] Mpu-9250 data sheet
[34] https://www.mathworks.com/videos/sensor-fusion-part-3-fusing-a-gps-and-an-imu-to-estimate-pose-1569911082630.html
[35] https://en.wikipedia.org/wiki/Kalman_filter
[36] https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications
[37] https://www.youtube.com/watch?v=CaCcOwJPytQ
[38] Youngjoo Kim and Hyochoong Bang (November 5th 2018). Introduction to Kalman Filter and Its Applications, Introduction and Implementations of the Kalman Filter, Felix Govaers, IntechOpen, DOI: 10.5772/intechopen.80600. Available from: https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications
[39] https://pythonrobotics.readthedocs.io/en/latest/modules/localization.html
[40] http://robotsforroboticists.com/kalman-filtering