Faculty Of Engineering

Self-Driving Car

Supervised by: Dr. Hesham Keshk

Team Members:

- **Ashraf Mohamed Mohamed Abdelmonem**
- **Fady Ibrahim Hana Kalds**
- **Muhammed Gamal El-Din Abdullah AbduAlkader**
- **Ahmed Hesham Mohamed Shawky**
- **Sara Yasser El-Sayed Ali**

Contents

Introduction:

The main goal of **autonomous vehicles** is to function properly without human assistance. To function, these cars use a combination of sensors, cameras, radar, and artificial intelligence (AI) to travel between destinations without a human operator.

Driverless cars use four components for working as follow:

1. Computer Vision

As human drivers, we need to be able to see the environment around us, whether it's looking ahead or reading street signs.

Computer vision is the way a car views its surroundings to identify objects in the car's vicinity. We do this by using an image classification network called Convolutional Neural Networks.

2. Sensor Fusion

Suppose the driver is stuck in heavy traffic. The driver knows this from the analysis of the environment, which corresponds to the idea of sensor fusion: Sensor fusion is one of the **most important components of self-driving vehicles**.

3. Localization

Self-driving car doesn't know where or how to get from point A to point B. The location works like a GPS system on a phone in autonomous cars. Used to keep track of locations.

4. Path Planning

Route planning is used to determine the best route to take. **Self-driving vehicles** use complex algorithms to get optimal routes.

Self-driving cars provide safe way to commute, significantly reduce traffic accidents. As 90% of accidents are caused due to human error, a recurring problem is that concerning the traffic, the driver is a human and he may miss some of the traffic signs on the road, so the car must be able to correct this error by recognizing and identifying in real time the traffic signs on the road. this feature makes the commute more safety and reduce traffic accidents and congestion.

Traffic accidents also due to distracted drivers Adaptive Cruise Control feature makes road crossing safer for the passengers it provides readings of front distance continuously and take suitable actions to slow down the car, stop it or speed up again.

If any object cut through the car road it is not going to stop for a long time instead it will take a safe action to avoid that object which will help us reduce the time taken to destination. Also increases productivity [The average one-way commute in the United States increased to a new high of 27.6 minutes in 2019] this means we are wasting approximately half a year commuting in our entire life.

Also, all of us having a problem to find a space to park this usually costs a lot of time to find a good space. After that Unprofessional driver are facing a hard challenge to perform the parking action by the help of self-parking feature, we do not have to waste time searching for the right place to park or worry about the action of parking as the parking action is not easy for the new drivers.

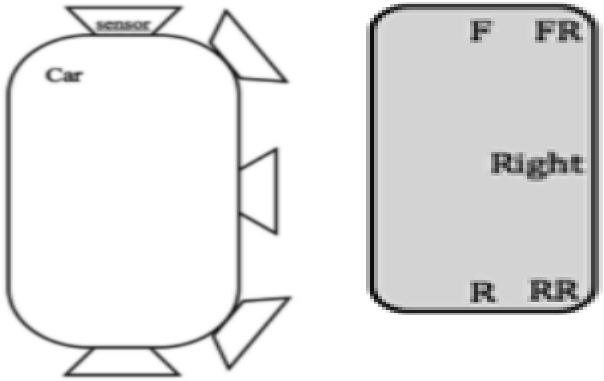
Another problem is knowing the way from our position to the destination also we don't know the traffic in this way the solution for the problem is using GPS feature it automatically find your position and find you the best way to take to achieve your destination.

(1) Self-parking Feature

1. Literature review.

An article was published by **Claire Chen** about **Autonomous Parallel Parking Car** winter 2015 he succeeds to park using the following logic. **[1]**

- **At first**, he used 5 sensors and place them as shown.



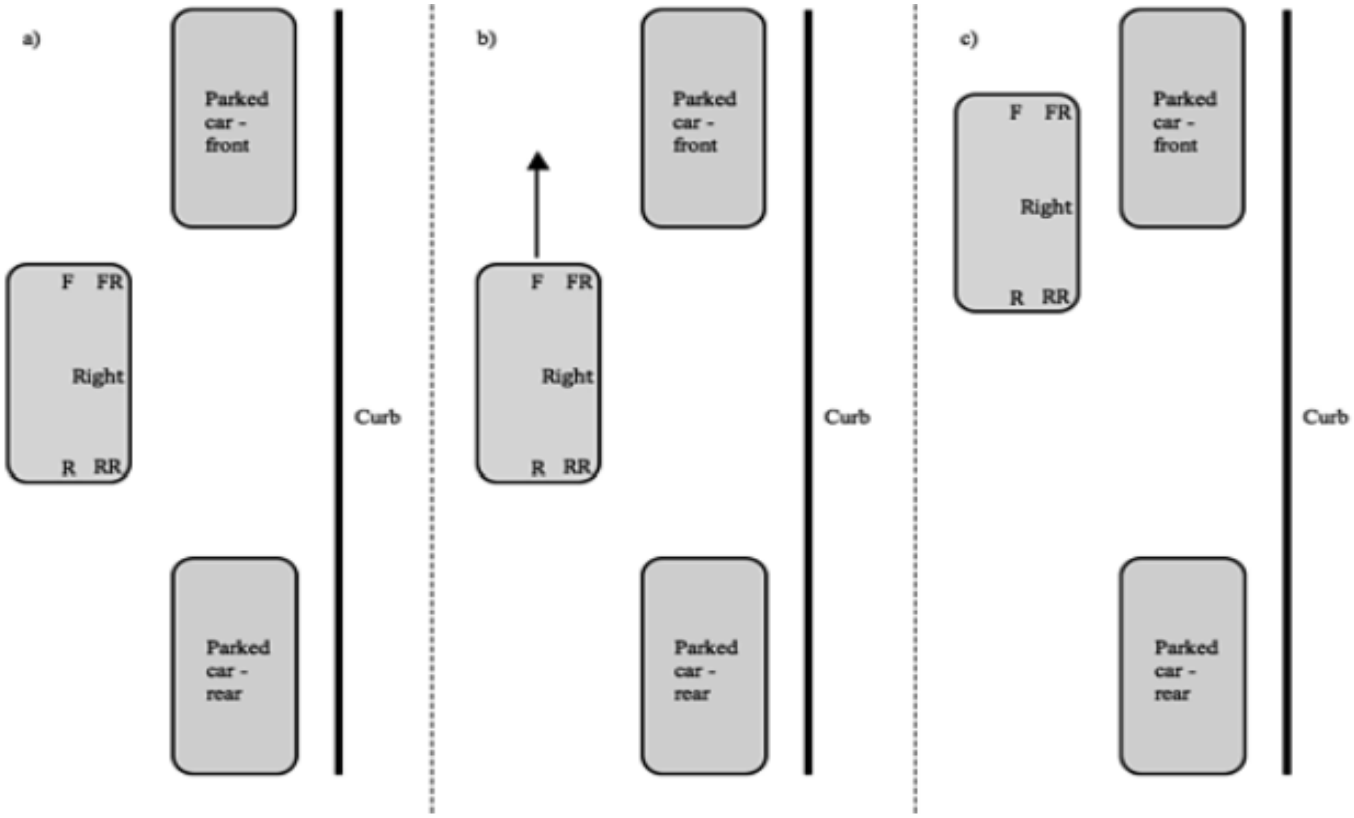
He also used a servo motor to control the front wheels and 2 DC motors for back wheels.

- **Motion 1: Finding a Parking Space.**

a) Pre-condition: The car begins on the left side of both the parked cars and parallel to the curb. The front of the parking car is in between both parked cars.

b) Code execution: Car begins moving straight forward. When right sensor and front right sensor are registering distances less than 9cm, car stops.

c) Post-condition: Car is stopped next to front car as shown.



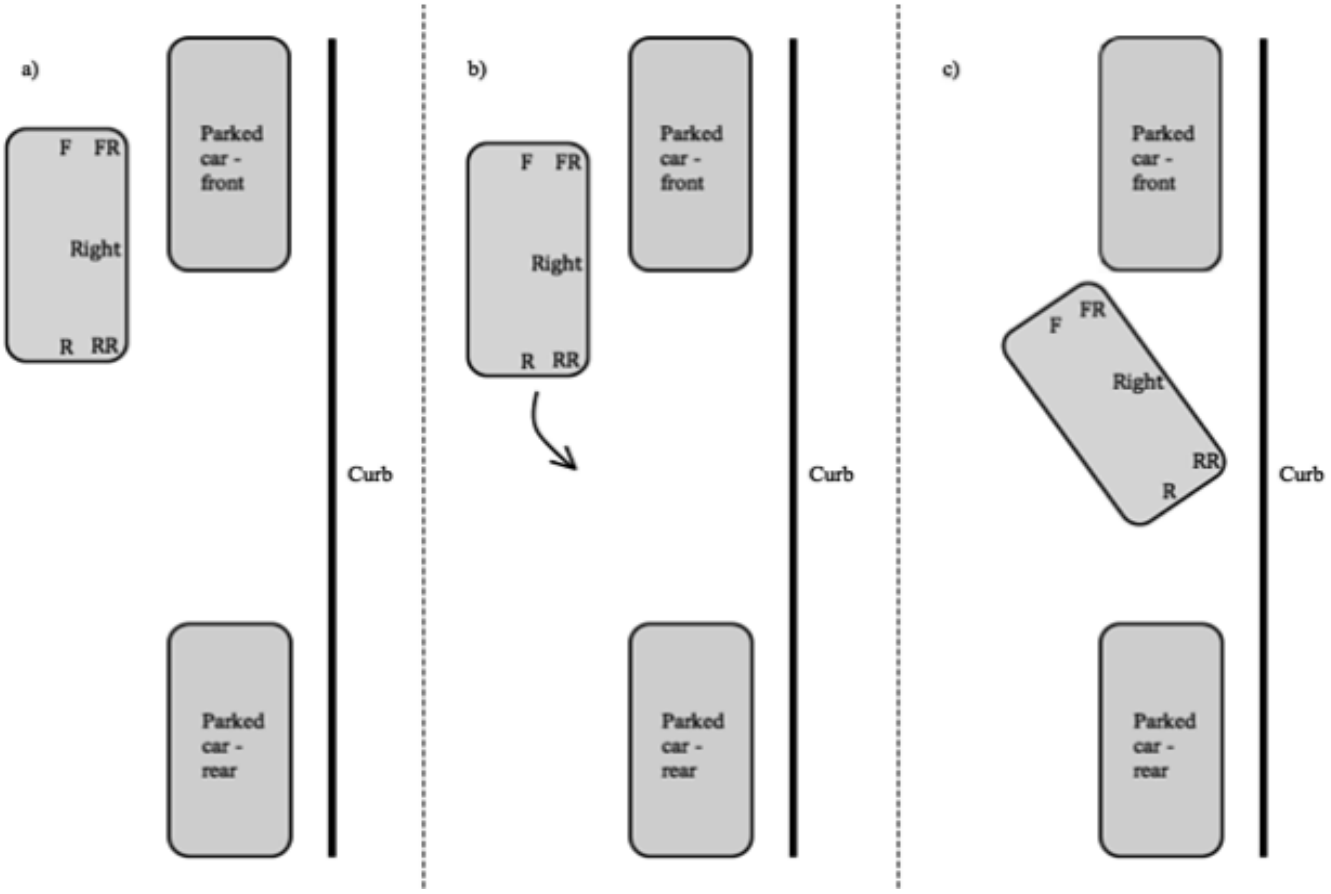
- **Motion 2: Reversing Toward Curb.**

a) Pre-condition: Post-condition of motion 1.

b) Code execution: Front wheels turn all the way to the right.

c) Car reverses at this angle until rear-right corner is within 7cm of curb.

d) Post-condition: Car is stopped at angle with rear-right corner close to curb as shown.



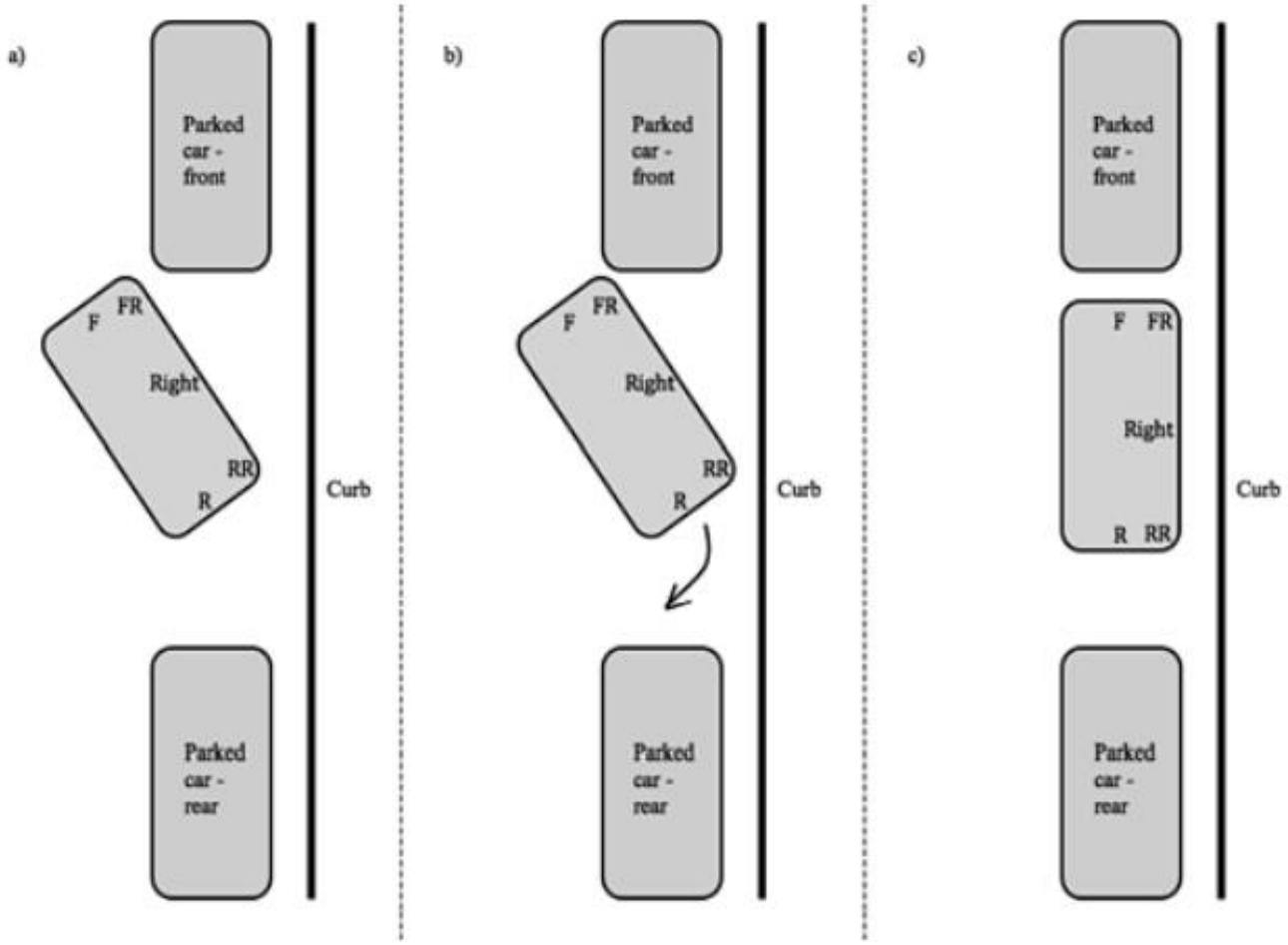
- **Motion 3: Straightening Out**

a) Pre-condition: Post-condition of motion 2.

b) Code execution: Front wheels turn all the way to the left.

c) Car reverses at this angle until rear sensor is within 4cm of car parked in the rear.

d) Post-condition: Car is stopped parallel to the curb and 4cm away from rear car.



Drawbacks: finding a park space algorithm **is not efficient** as if the distance between the parked cars is small, he can't detect that, also he used **a lot of sensors** to achieve parking goal.

Hint: he uses Arduino uno to control the process.

---

2. Modules used and their number.

In this project parallel parking is achieved using **3 ultrasonic sensors**, **2 DC Motors**, **Motor driver l298N**, **a Servo Motor**, **The LM393 IR Module, A wheel encoder** and **ATmega32 Microcontroller**.

Ultrasonic sensor:

- **HC-SR04 Hardware Overview**

  An HC-SR04 ultrasonic distance sensor consists of two **ultrasonic transducers.**

  One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses. When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front. **[2]**

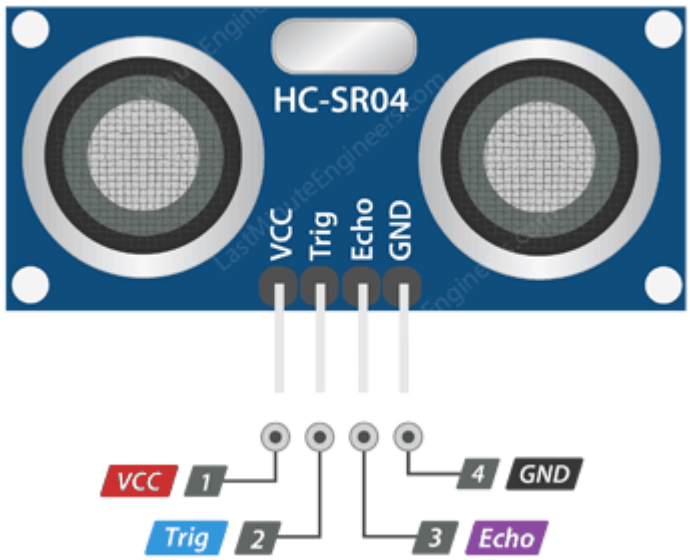
  This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm.

  Hint: maximum range is **4m theoretically** practically **80cm**.

- **Technical Specifications**

Operating Voltage	DC 5V
Operating Current	15mA
Operating Frequency	40KHz
Max Range	4m
Min Range	2cm
Ranging Accuracy	3mm
Measuring Angle	15 degree
Trigger Input Signal	10µS TTL pulse

- **HC-SR04 Ultrasonic Sensor Pinout**



VCC supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino.

Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to high for 10µs, the sensor initiates an ultrasonic burst.

Echo pin goes high when the ultrasonic burst is transmitted and goes low. By measuring the time, the Echo pin stays high, the distance can be calculated.

- **How Does HC-SR04 Ultrasonic Distance Sensor Work?**
  - It all starts when the trigger pin is set HIGH for 10µs. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise.

  - These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal.

  - If those pulses are not reflected, the echo signal times out and goes low after 38ms (38 milliseconds). Thus, a pulse of **38ms** indicates no obstruction within the range of the sensor.

- **Calculating the Distance**

  The width of the received pulse is used to calculate the distance from the reflected object. This can be worked out using the simple distance-speed-time equation.

Distance = Speed x Time

Hint: The approximate speed of sound in dry (0% humidity) air, in meters per second, at temperatures near 0 °C, can be calculated from $c_{\text{air}} = (331.3 + 0.606 \cdot \theta) \quad \text{m/s},$

where is **theta** the temperature in degrees Celsius (°C).

So, the speed of the sound is function in temperature.

- ## DC Motor and Motor driver l298N:

  ### Controlling a DC Motor

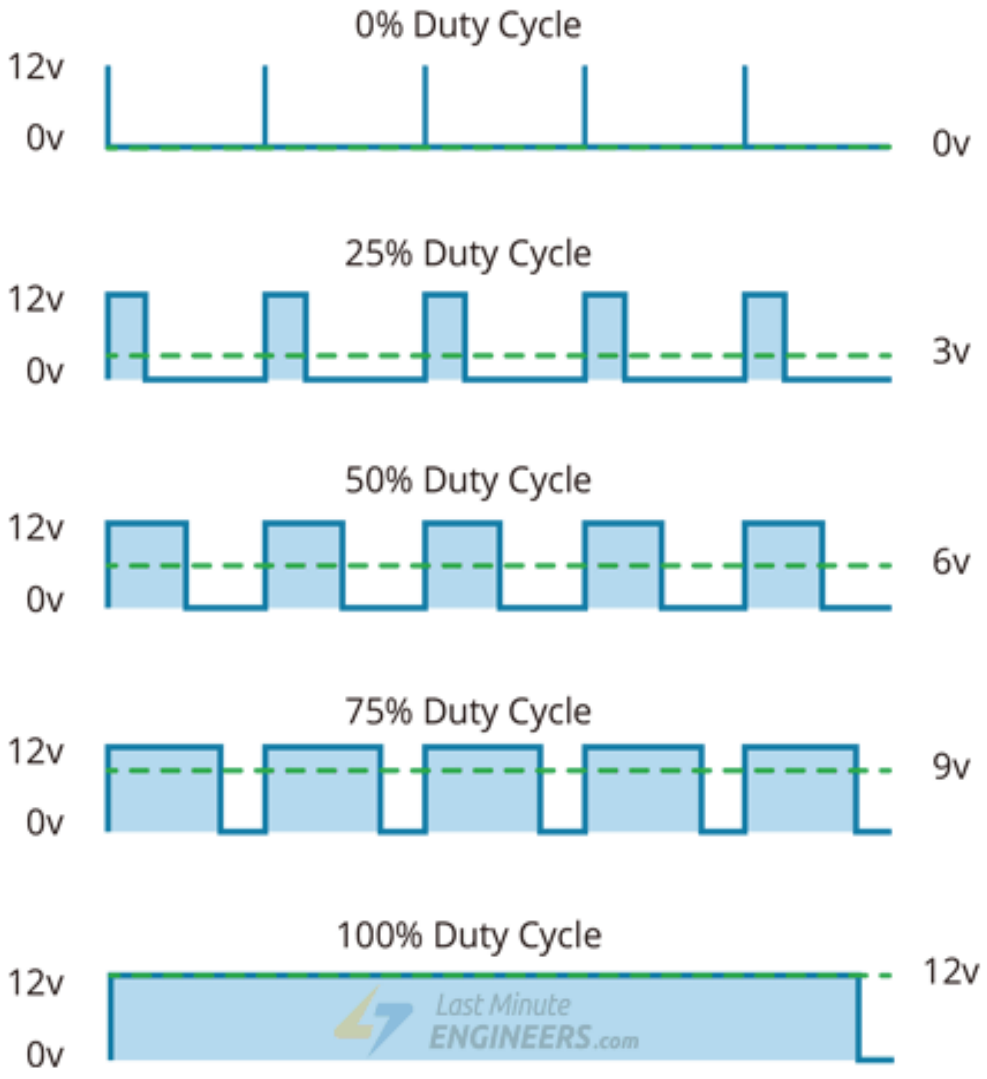
  To have complete control over DC motor we must control its speed and rotation direction. This can be achieved by combining these two techniques. **[3]**

  1.PWM – to control speed

  2.H-Bridge – to control the rotation direction

1.PWM – to control speed

- The speed of a DC motor can be controlled by changing its input voltage.
- A common technique to do this is to use PWM (Pulse Width Modulation).
- PWM is a technique where the average value of the input voltage is adjusted by sending a series of ON-OFF pulses.
- The average voltage is proportional to the width of the pulses known as the Duty Cycle.
- The **higher** the duty cycle, the higher the average voltage applied to the DC motor **(resulting in higher speed)** and the **shorter** the duty cycle, the lower the average voltage applied to the DC motor **(resulting in lower speed)**.
- The image below shows PWM technology with different duty cycles and average voltages.

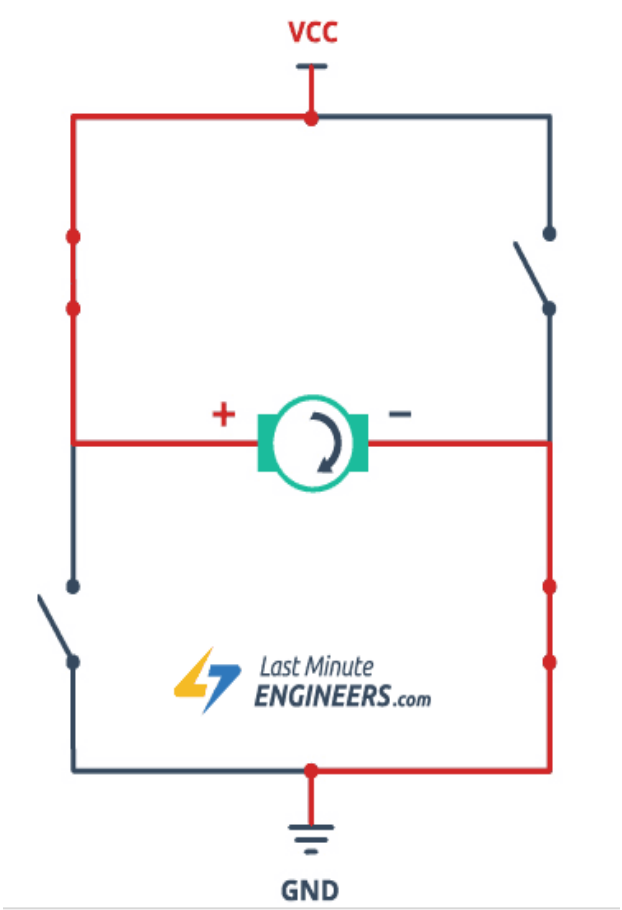


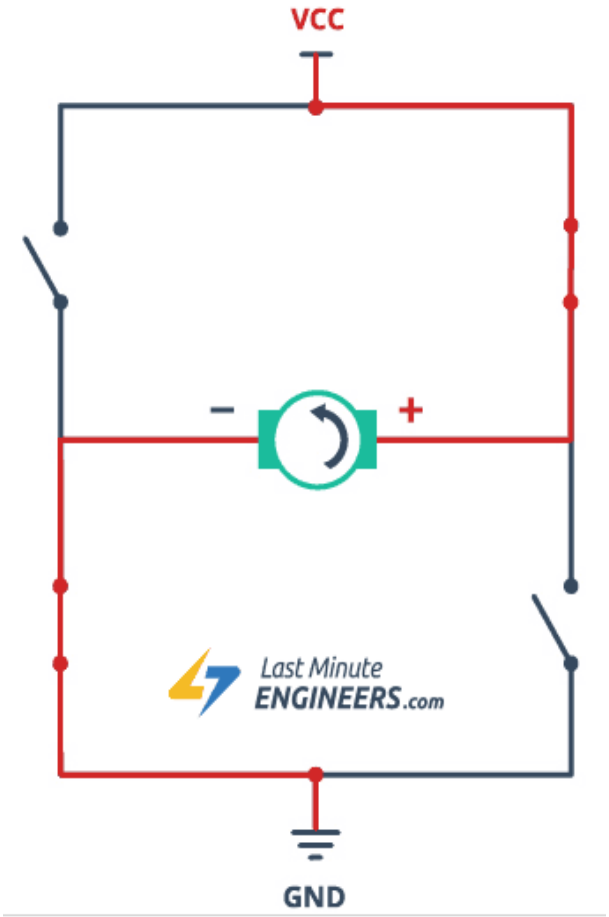
Pulse Width Modulation(PWM) Technique

2.H-Bridge – to control the rotation direction

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-bridge. An H-bridge circuit consists of four switches with the motor in the center forming an H-like arrangement.

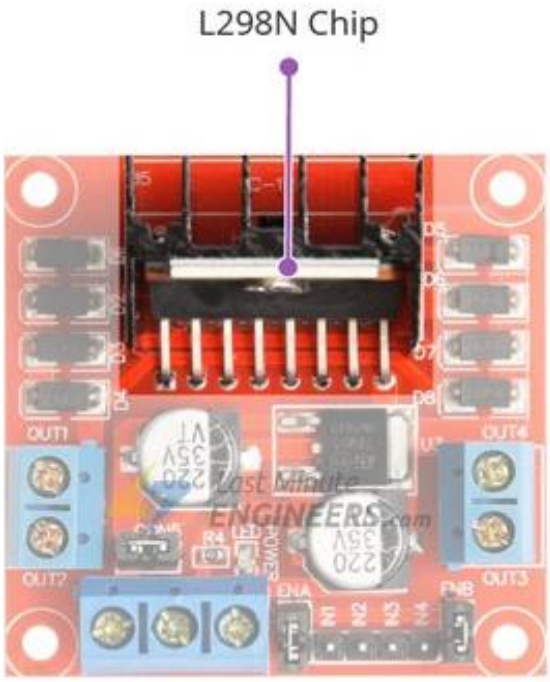
Closing two specific switches at a time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.





- **L298N Motor Driver Chip:**

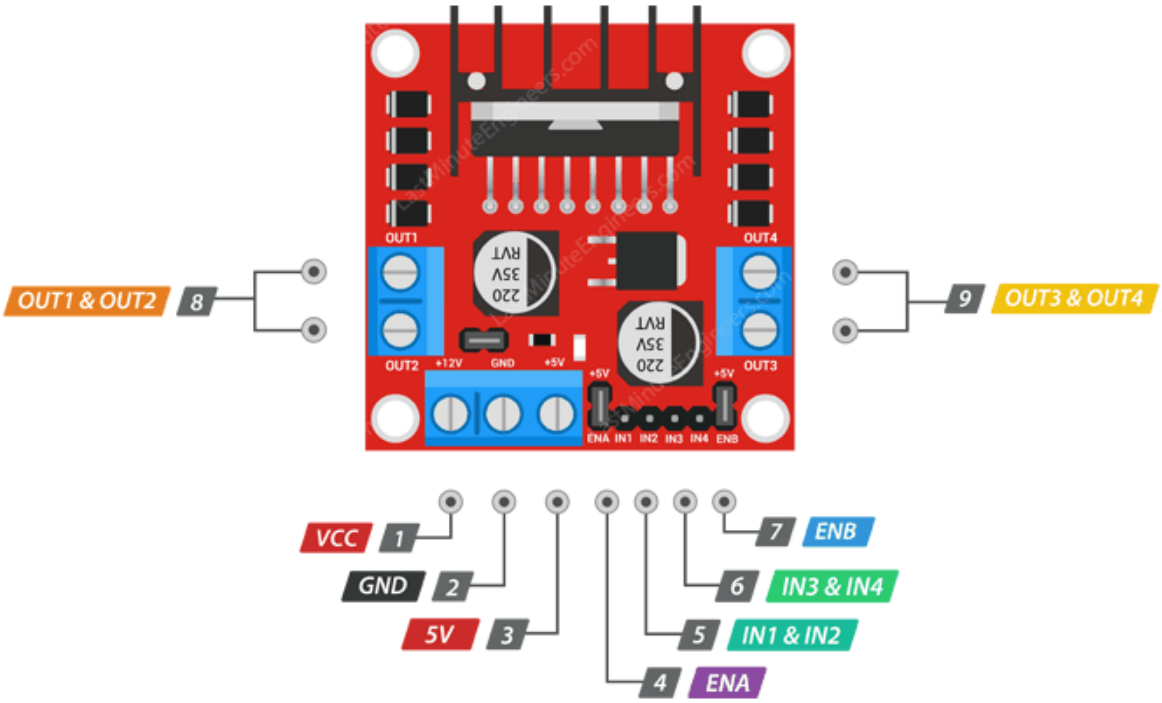
At the centre of the module is a big, black chip with a chunky heat sink – the L298N.



Inside the L298N chip, you'll find two standard H-bridges capable of driving a pair of DC motors. This means it can drive up to two motors individually which makes it ideal for building a two-wheeled robotic platform.

The L298N motor driver has a supply range of 5V to 35V and is capable of 2A continuous current per channel, so it works very well with most of our DC motors.

- **L298N Motor Driver Module Pinout:**



1. VS pin gives power to the internal H-Bridge of the IC to drive the motors. You can connect an input voltage anywhere between 5 to 12V to this pin.
2. VSS is used to drive the logic circuitry inside the L298N IC which can be 5 to 7V.
3. GND is the common ground pin.
4. The L298N motor driver's output channels OUT1 and OUT2 for motor A and OUT3 and OUT4 for motor B You can connect two 5-12V DC motors to these terminals.
5. The module has two direction control pins for each channel. The IN1 and IN2 pins control the spinning direction of motor A, While IN3 and IN4 control the spinning direction of motor B. The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs. The chart below shows how this is done.

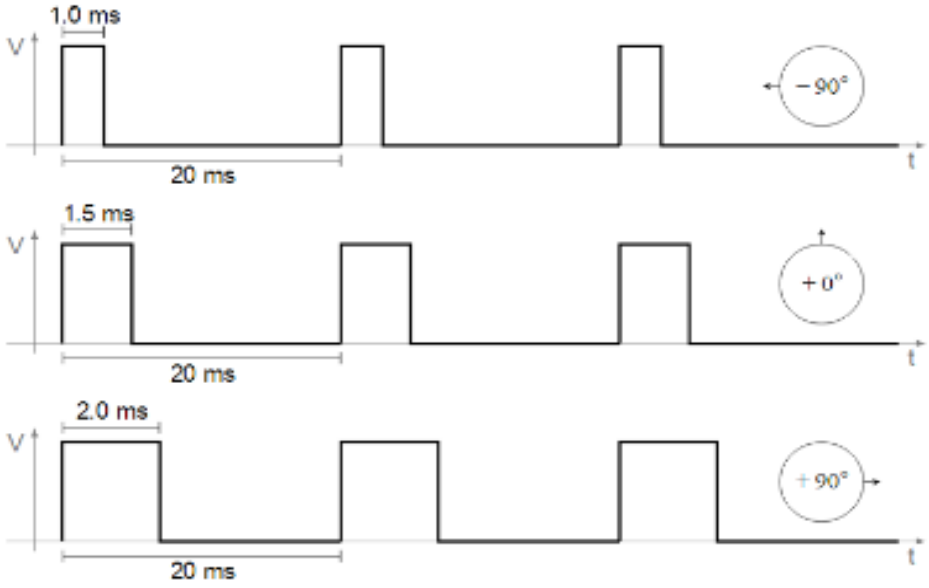
Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

6. The speed control pins ENA and ENB are used to turn on/off the motors and control its speed. Pulling these pins HIGH will cause the motors to spin, while pulling it LOW will stop them. But, with Pulse Width Modulation (PWM), you can control the speed of the motors.
7. The module has an on-board 5V regulator – 78M05. It can be enabled or disabled via a jumper. When this jumper is in place, the 5V regulator is enabled, which derives the logic power supply (VSS) from the motor power supply (VS). In this case, the 5V input

terminal acts as the output pin and delivers 5V 0.5A. You can use it to power **servo motor**.

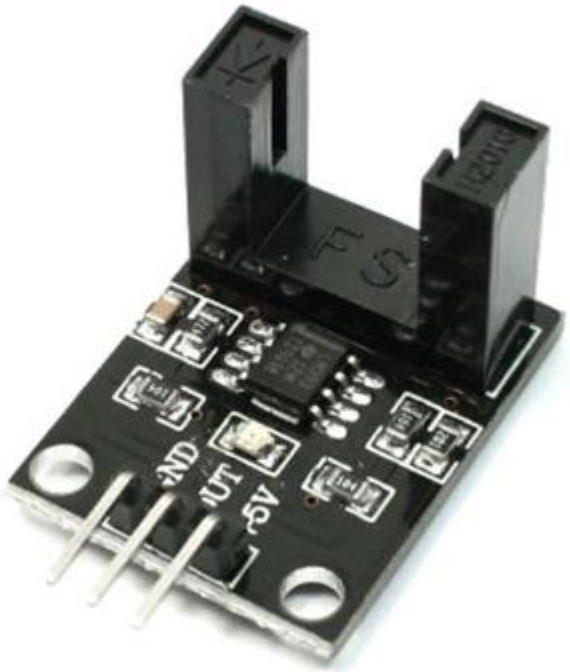
- **Servo Motor**

1. Usually, a servomotor turns 90°in either direction, i.e., maximum movement can be 180°.
2. A normal servo motor cannot rotate any further due to a built-in mechanical stop.
3. Three wires are taken out of a servo: positive, ground and control wire.
4. A servo motor is controlled by sending a Pulse Width Modulated (PWM) signal through the control wire. A pulse is sent every 20 milliseconds.
5. Width of the pulses determine the position of the shaft.
6. For example, a pulse of 1ms will move the shaft anticlockwise at -90°, a pulse of 1.5ms will move the shaft at the neutral position that 0° and a pulse of 2ms will move the shaft clockwise at +90°.



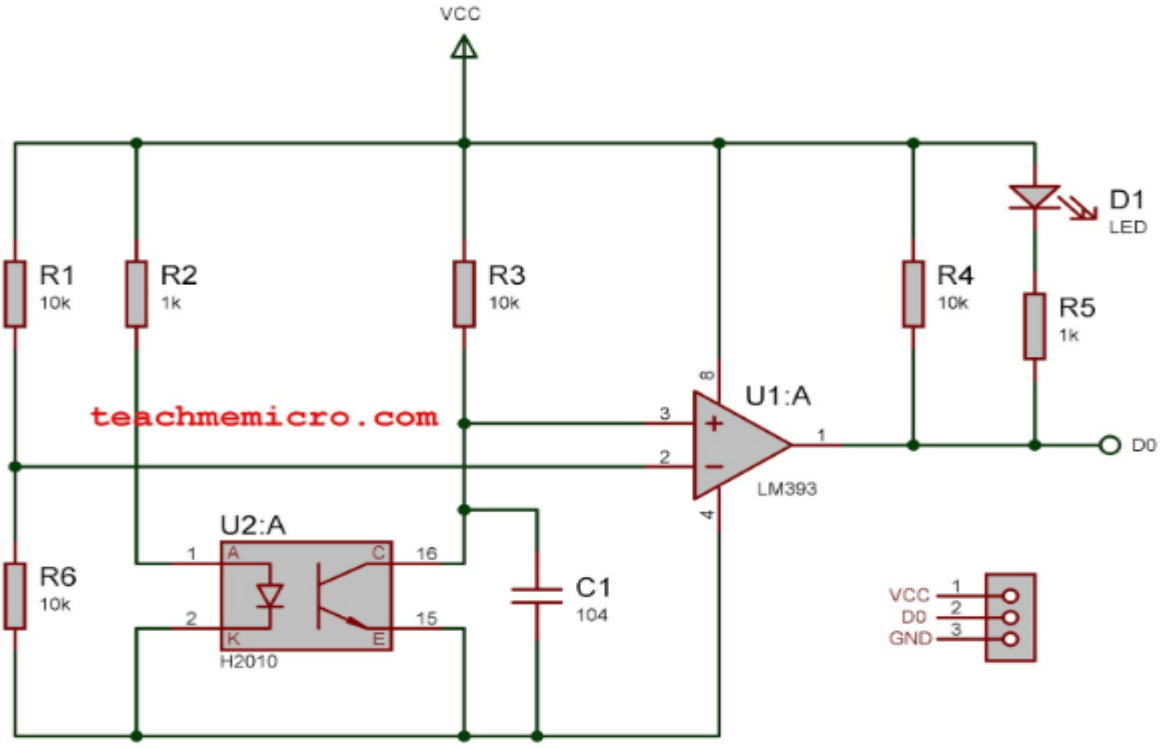
7. When servo motor is commanded to move by applying pulses of appropriate width, the shaft moves to and holds the required position. If an external force is trying to change the position of the shaft, the motor resists to change. Pulses need to be repeated for the motor to hold the position.

- **The LM393 IR Module**



The module has two vertical columns with an IR LED on one column and a phototransistor on the other.

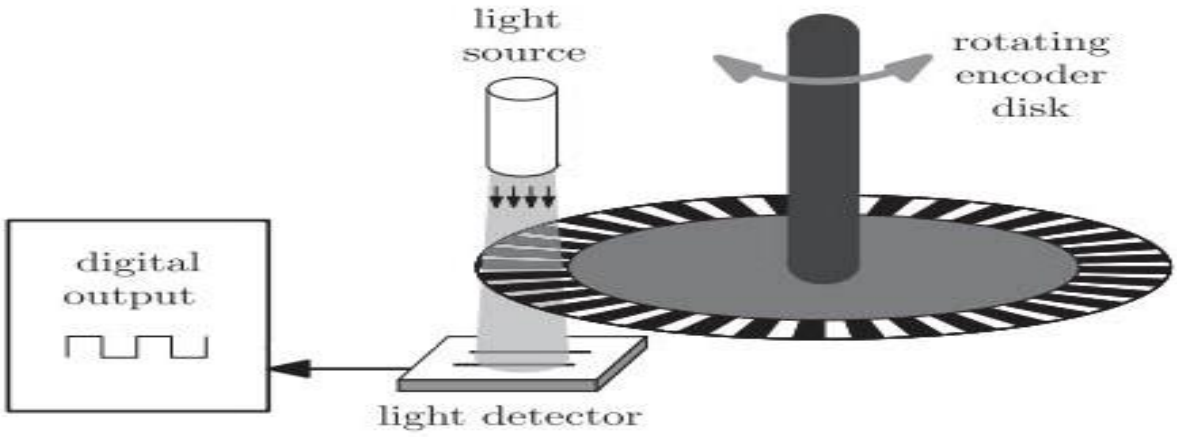
Whenever the path between the IR LED and phototransistor is cut, the **output pin(D0) goes high**.



Here we see how simple the circuit for the module is. With no obstruction between the IR LED and phototransistor, the voltage between the positive and negative terminals of the comparator is equal. When the phototransistor is blocked, it will draw a higher voltage, making the positive terminal of the comparator more positive than the negative terminal. Thus, a positive voltage, equal to VCC, will be at the D0 terminal.

- **A wheel encoder**

With proper placement, the slots on the encoder will block or pass the IR LED. This will create a train of pulses whose frequency is proportional to the speed of the motor.



The wheel encoder pictured has 20 slots. Therefore, counting 20 pulses mean the wheel has traveled one revolution.

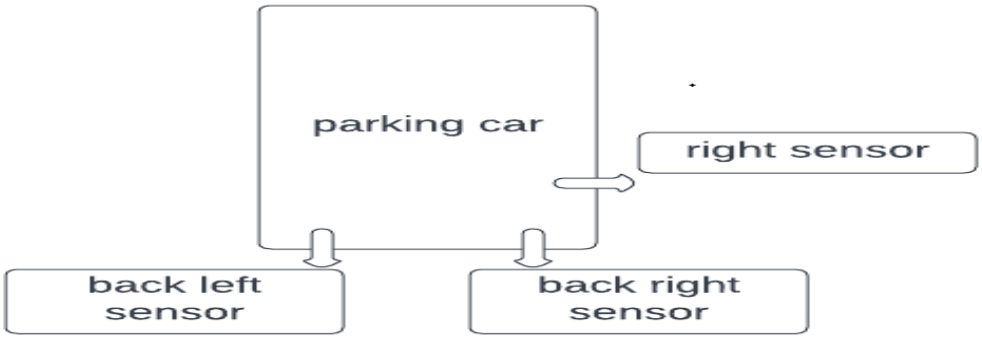
HINT: The diameter of the wheel is 65 mm.

Circumference is: 3.14 * diameter→ 3.14 * 65=204.1 mm =204.1/10 **=20.41cm**

Therefore, each one revolution means we move 20.41 cm.

3.Parking logic algorithm.

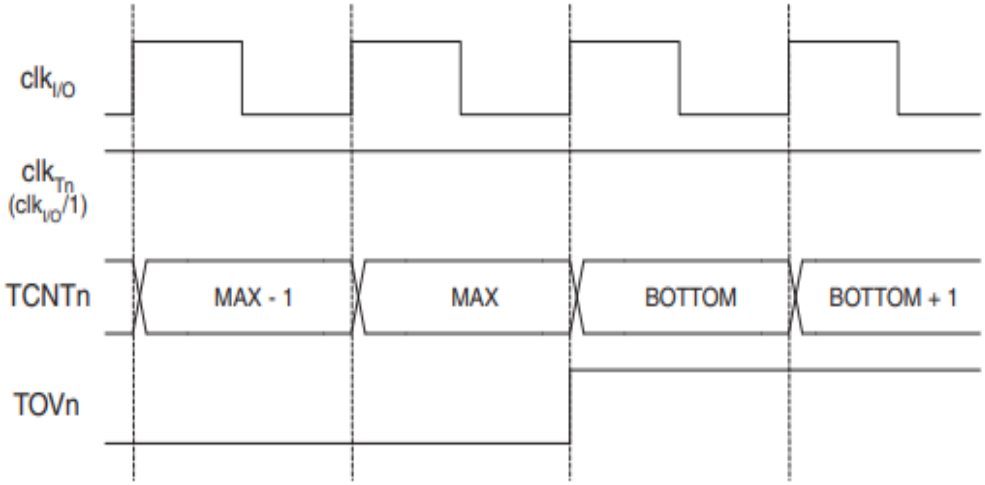
To achieve parking goal in our project 3 ultrasonic sensors are used and positioned as shown:



Phase 1

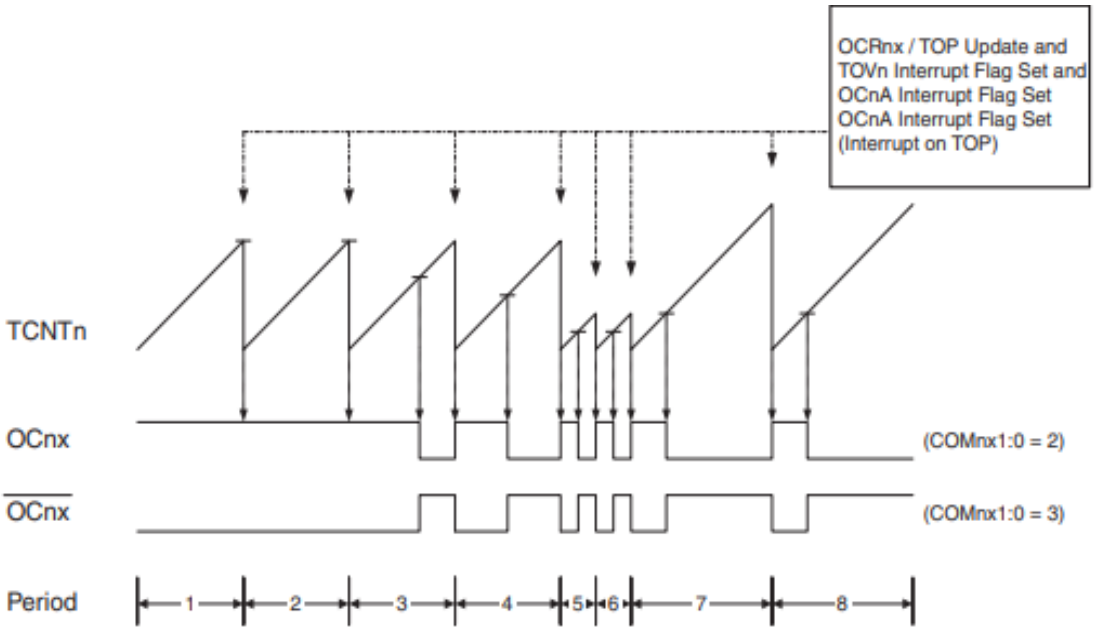
- Here we set timer 0 in **overflow mode** to count the time.

Figure 34. Timer/Counter Timing Diagram, no Prescaling



The program is working with internal RC frequency of 1MHz as we are working without pre-scaling the timer tick time which means the time to increase its value in TCNT0 by 1 is 1/F_CPU=1/1MHz therefore its tick time is 1 microsecond.

- Timer one is set to work in Fast PWM Mode (Non-inverting) we will configure it to count to the value in **ICR1 REGISTER** we will load it by 20,000 to make the time period of the pulses 20ms **as our tick time is 1micro second** this will help us to control Servo Motor by changing the value in **OCR1A** we can produce a signal with 20ms with different duty cycles.

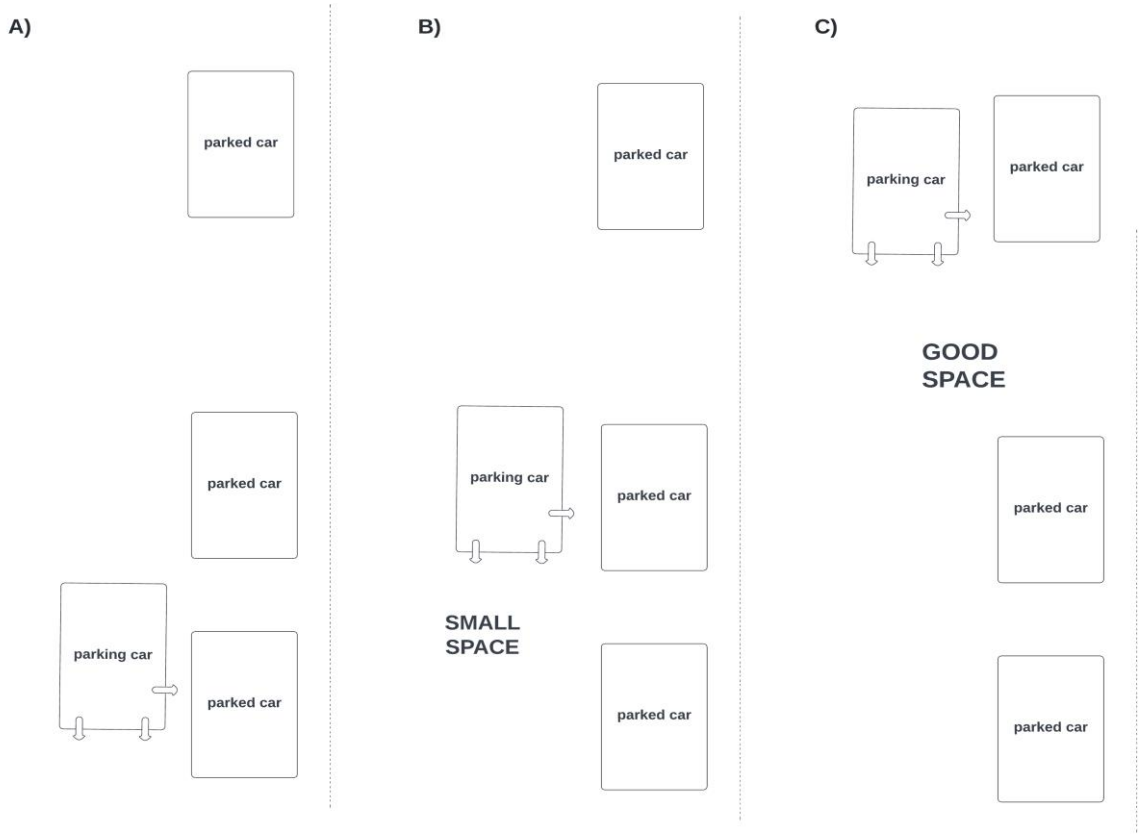


- Timer 2 is set to work in overflow mode and with the help of interrupt we can detect rising and falling edge of the echo pin from ultrasonic and we can count the time of pulse from that we can get the value of the distance between the ultrasonic and the object.

- **Motion 1: Finding a Parking Space.**

1. In this stage the car start moving forward while maintaining the servo motor to 0 degree.
2. If the right sensor registers a distance greater than 12, we enable one of the timers to work in overflow mode to count the time.

3. If the right sensor registers a distance less than 12 the car stops next to the front parked car.
4. Then we start calculating the distance from the equation d=v*t
5. If the distance is small the car ignores it and moves forward to find another space to park.



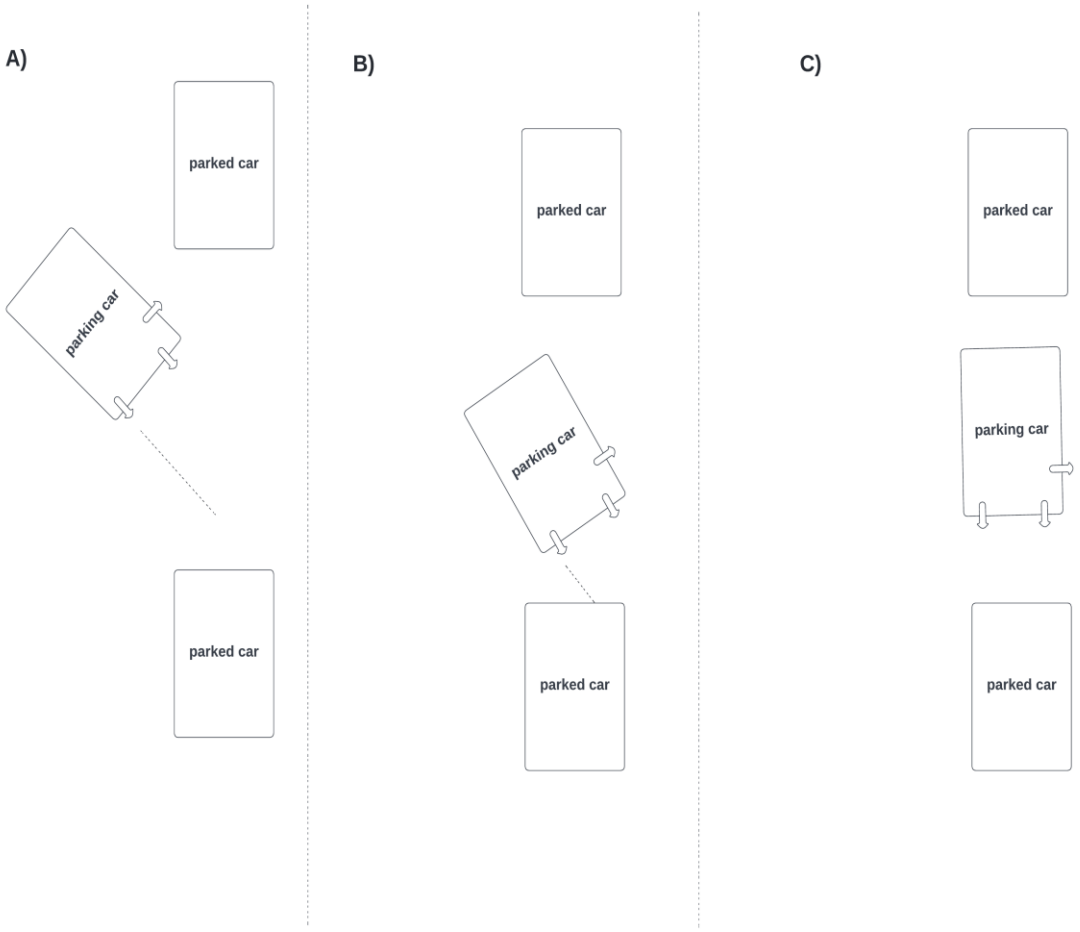
- **Motion 2: Taking Parking action.**

1. The front wheels turn right 57 degrees and the car start moving backward while back right sensor registers a distance

   - greater than 60 cm
   - then smaller than 60 cm
   - and value another greater than 60 cm

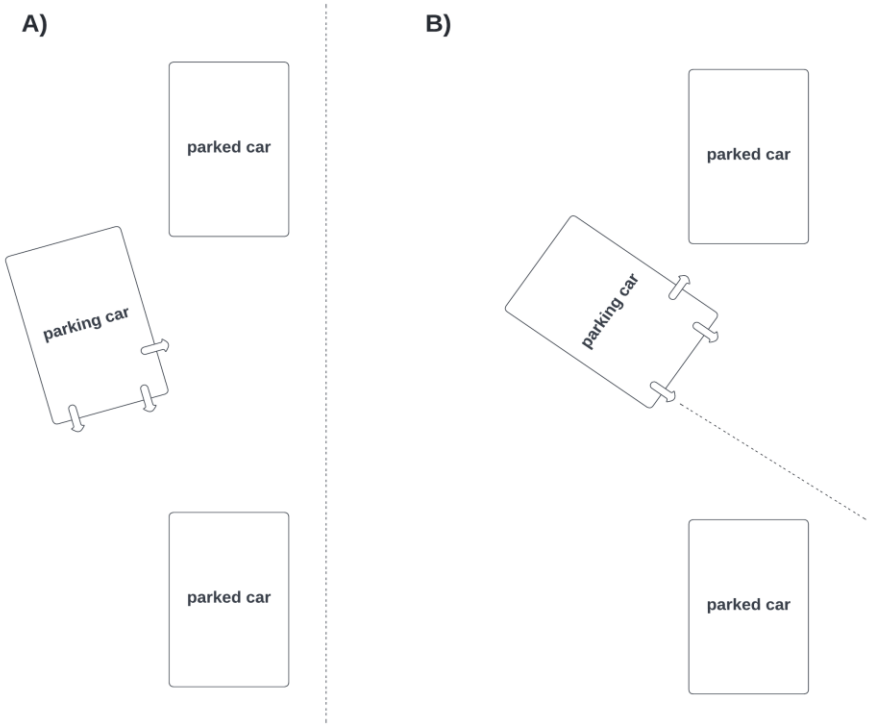
2. Then the wheels turn to zero degree and the car move backward and taking values from back left sensor Until it registers a distance of 5 cm then the car stops

3. Then the wheels turn to 55 degree and start moving forward until **back left sensor registers 10 cm** then the car stops 10 cm forward the back car.



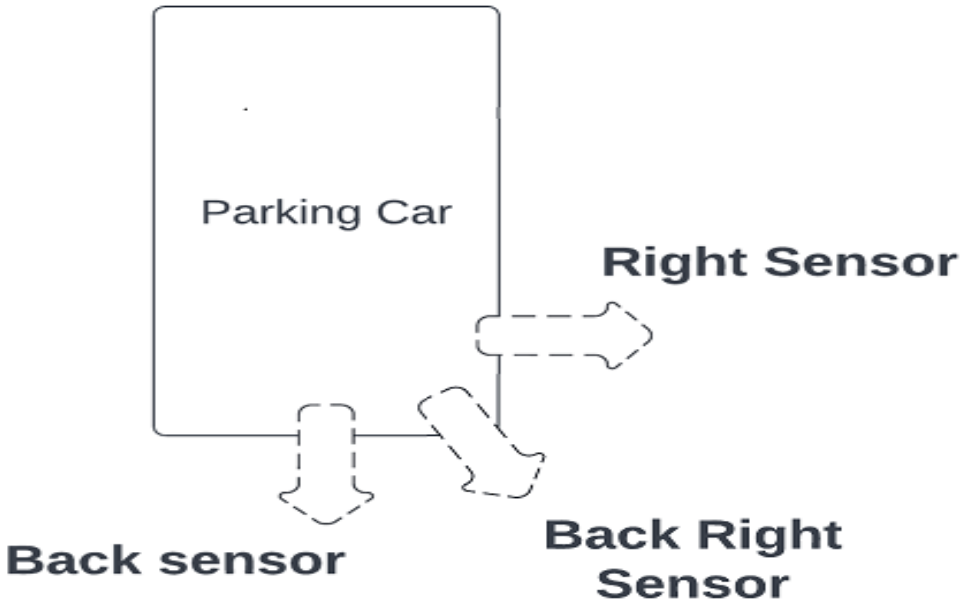
Most of times the back left sensor loses the range

- At this point the front wheels turn to left with 60 degree and start moving backward until getting in the range then performing the action mentioned above.



Phase 2

Sensor's position



- Here we are not going to calculate the distance by this equation **d=v*t** instead we are going to count the number of pulses generated from the **IR MODULE** we can do that by the help

of interrupt which count the number of times the output pins of IR Module go high form that we can detect the distance as we mentioned in the **wheel encoder part**.

- Here we are going to set **Timer 1** to work in Fast PWM Mode as **Phase1** but the top here is going to be **OCR1A Register** we will load it with 20,000 to generate 20ms signal to control Servo Motor the difference here is that we are going to use **Timer1** also to help us work with ultrasonic sensors by enabling the **input capture unit** we can detect the rising and falling edges generated by ultrasonic on its echo pin.

1. We are going to put the value of the counter say in **rising_edge** variable if we detect rising edge.
2. the timer still counts until falling edge similarly we are going to put its value at falling edge in **falling_edge** variable.
3. form the difference between them **falling_edge - rising_edge** we can have the duration of the pulse from that we can find the distance between the ultrasonic and the object.
4. The case of falling edge value may be smaller than rising edge value is considered (case timer overflows after taking rising edge value)

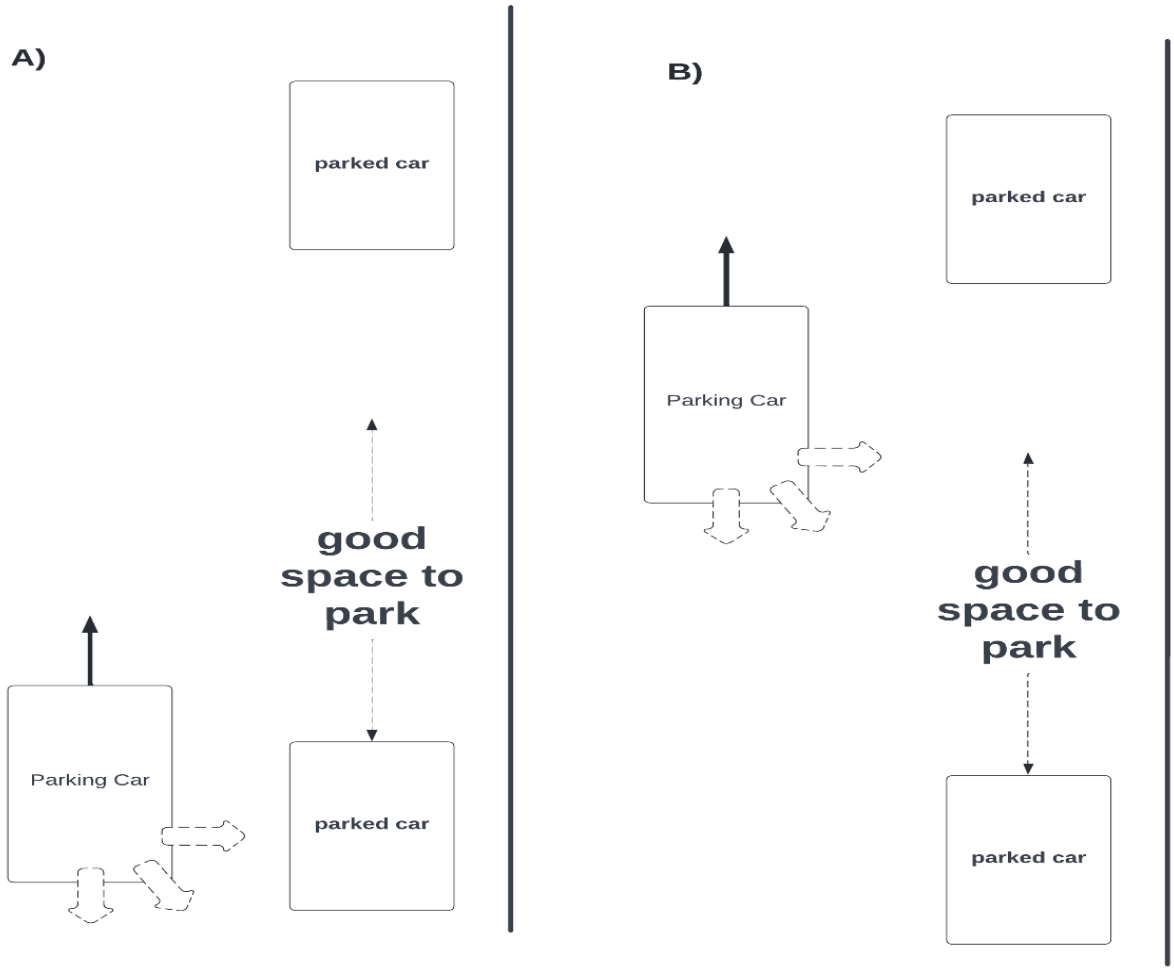
    - By subtracting the rising edge from the maximum value stored in OCR1A
    - Then adding that value to the falling edge value
    - Ex. Falling_edge=80, Rising edge=19,000
    - Calculations = ((20,000-19000) +80) *timer tick time

Here the new logic of parking

- **Motion 1: Finding a Parking Space.**

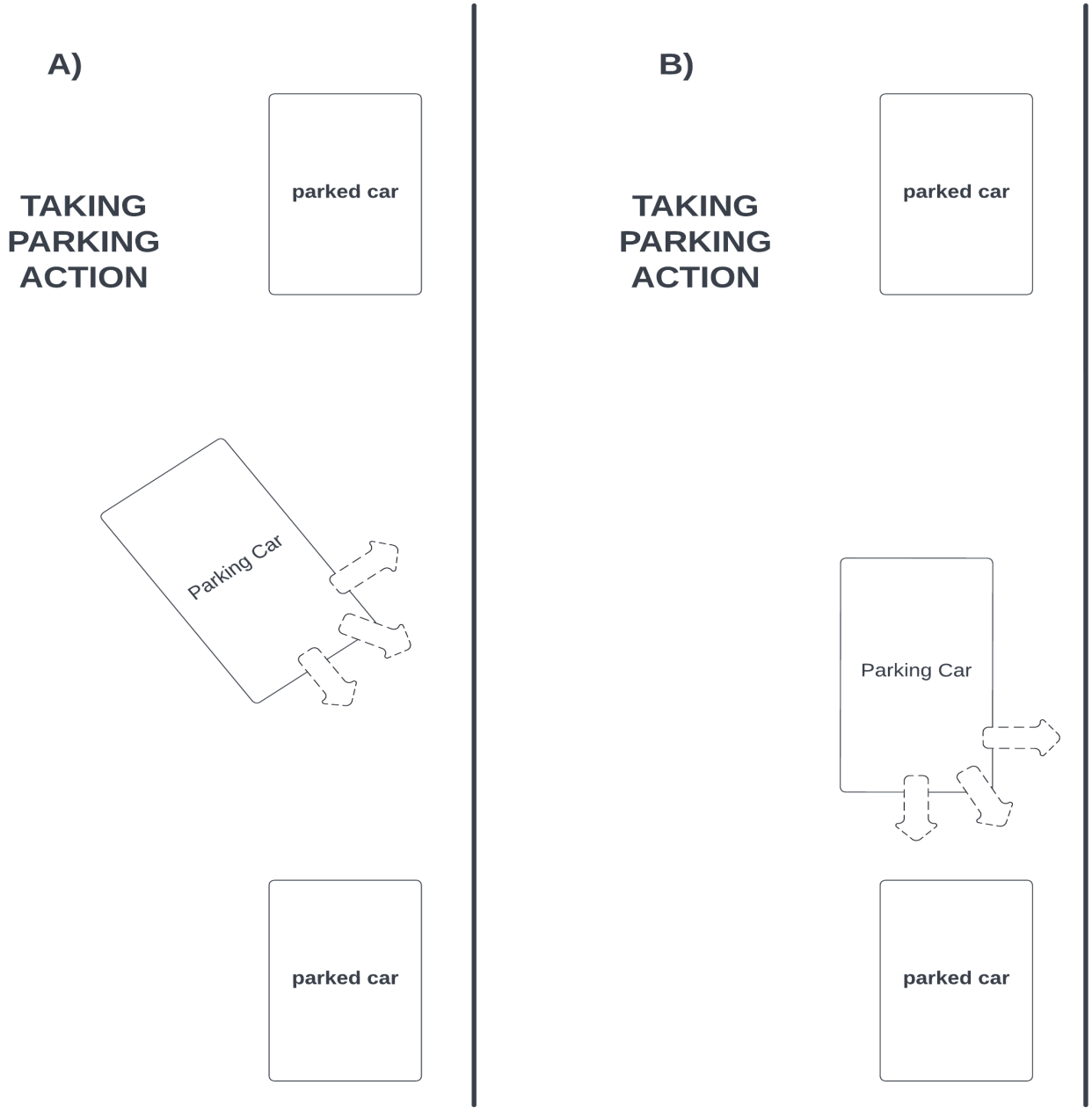
    1. In this stage the car start moving forward while maintaining the servo motor to 0 degree as the right sensor registers a distance less than 12.
    2. If the right sensor registers a distance greater than 12, we enable the interrupt to count the pulses from the IR Module.

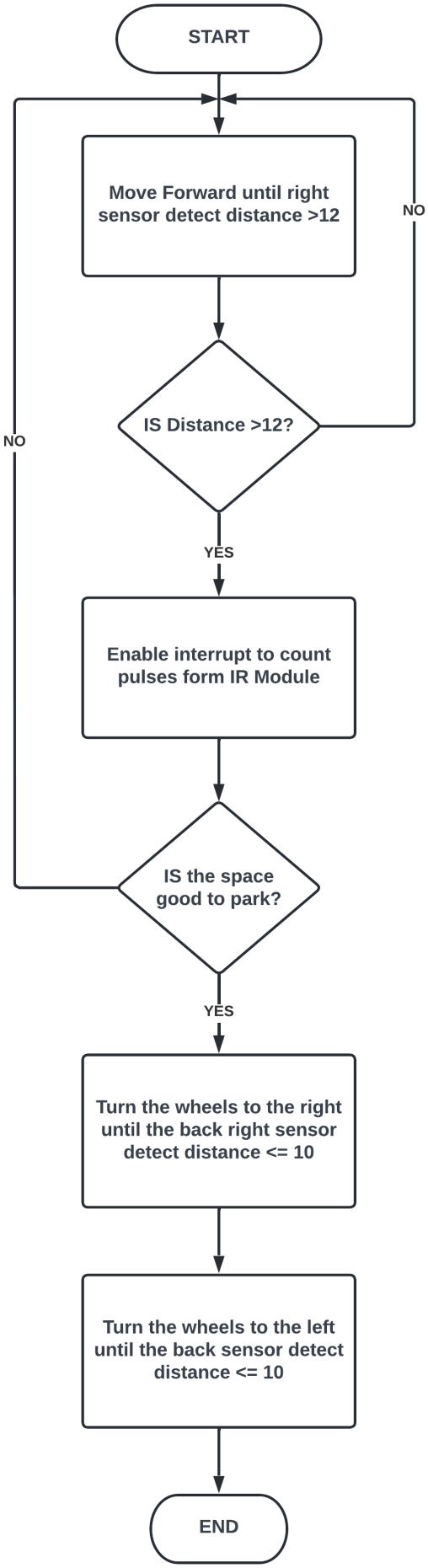
3. Here if we count number of pulses corresponding to sufficient distance **we don't need to keep moving** until detecting the front car **we can take the action of parking directly**
4. If we detect the front car and we haven't reached to the sufficient number of pulses the car will detect that the distance is small, ignores it and moving forward to find a good one



- **Motion 2: Taking Parking action.**

1. The front wheels turn right with 60 degree and start moving backward until the back right sensor detect distance smaller than or equal 15.
2. Then the front wheels turn left with 60 degree and the car move backward until the back sensor detect distance of 10cm
3. The car park front of the back parked car




START
Move Forward until right sensor detect distance >12
NO
IS Distance >12?
NO
YES
Enable interrupt to count pulses form IR Module
IS the space good to park?
YES
Turn the wheels to the right until the back right sensor detect distance <= 10
Turn the wheels to the left until the back sensor detect distance <= 10
END

- **<u>A very important thing to consider.</u>**

    a. Atmega32 have internally 3 timers and it can generate 4 PWM signals only

    b. As mentioned before PWM Signal is the core of the process we need:

- One to control the speed of the DC motors
- One to control servo motor
- And three of them to work with the ultrasonic sensors
- Therefore, we need **5 PWM signal** and that is not supported by Atmega32.

    c. So **CD4052 4-Channel Demultiplexer IC** is used as there is no need to work with many sensors at a time.

- **<u>CD4052 4-Channel Demultiplexer/Multiplexer IC:</u>**

CD4052 Pinout

Pin Configuration

Pin Number	Pin Name	Description
16	Vdd	Positive power input, maximum 20V
7	Vee	Negative power rail, normally connected to ground.
8	Vss (Ground)	Connected to ground of the circuit
6	INH	Enable pin – Must be pulled to ground for normal operation
9,10	A,B	Channel Select pins
1,12	Y0,X0	Channel 0 Input / Output
5,14	Y1,X1	Channel 1 Input / Output
2,15	Y2,X2	Channel 2 Input / Output
4,11	Y3,X3	Channel 3 Input / Output
3,13	Y,X	Common Output / Input

CD4052 as 1:4 Demultiplexer:

The **CD4052** can be used as a **1:4 Demultiplexer** also, that is it can take one input and provide it either one of the 4 output channels based on the channel select pins. Here the input pins will be X and Y. The output pins can either be X0, Y0 or X1, Y1 or X2, Y2 or X3, Y3 based on the value set on A and B pins.

A	B	Channel Selected
0	0	Channel 0
1	0	Channel 1
0	1	Channel 2
1	1	Channel 3

(2) Adaptive Cruise Control feature

- **Adaptive Cruise Control:** is feature aid to provide safe cruise for the passenger by always keeps safe distance which make the car slow down or stop safely without any crushes or loses in lives.

1- Distracting Drive.

In its simplest form, distracted driving includes any kind of activity that takes your attention away from the road when you are behind the wheel. While most people associate distracted driving with cell phone use — certainly a significant contributor — distracted driving encompasses numerous behaviors like using your phone calling someone or texting, turning around, carrying on a conversation with a passenger, using electronics, like a GPS or sound system and even eating while driving in some cases.**[5]**

2- Statics of accidents caused by distraction.

In April 2006, a study released by the Virginia Tech Transportation Institute and the NHTSA found that **almost 80% of crashes and 65% of near-crashes involved some form of driver attention within three seconds of the event.** The study goes on to find the most common distraction was phone use.

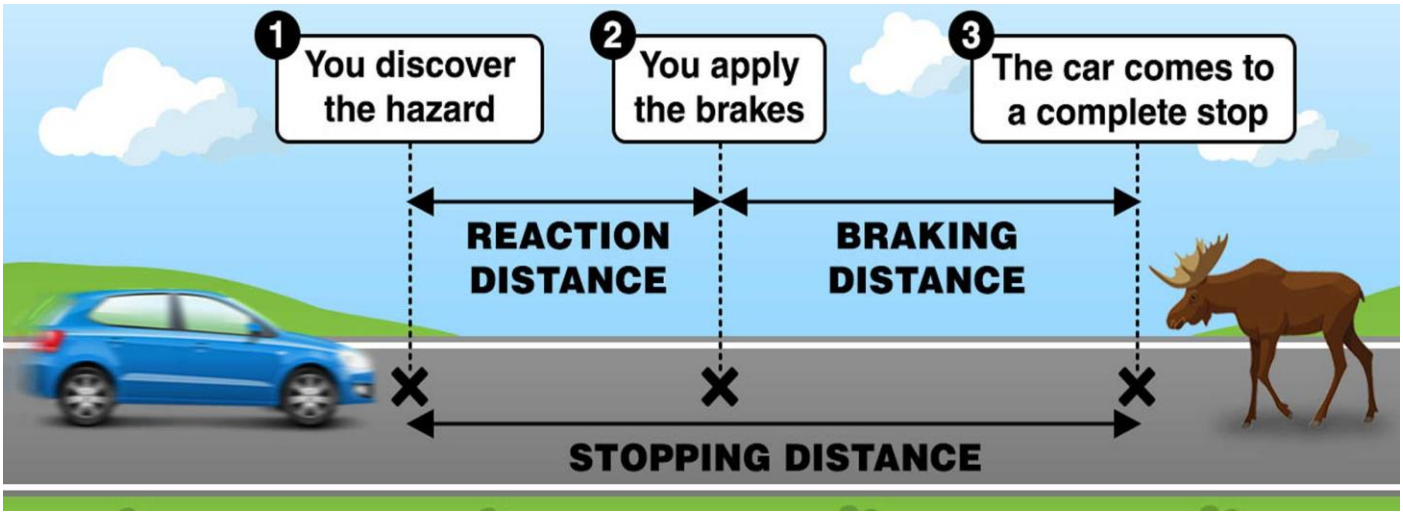
According to historical data from 2010 to 2018, the number of fatalities from distracted driving appears to have decreased. However, when you compare the number of distracted driving accidents in total to the overall number of crashes, distracted driving has remained a steady contributor, causing between 14% to 17% of all crashes as shown in table (1).

YEAR	Distracted driving deaths	Distracted driving accidents
2019	2895	986000
2018	2645	938000
2017	3003	912000
2016	3197	905000
2015	3242	885000
2014	2972	967000
2013	2910	904000
2012	3098	908000
2011	3047	826000
2010	2993	900000

So, if we could reduce the number of distracted driving, we will reduce the main number of accidents every year and save thousands lives. This feature will help so much to avoid crushes while distraction of the driver.

3- Problem Solution

We cannot prevent human errors or stop the driver's distractions, we can reduce them by raising awareness against the dangers of distraction while driving, but of course it cannot be completely prevented, so we need to develop new technology that helps solve this problem.

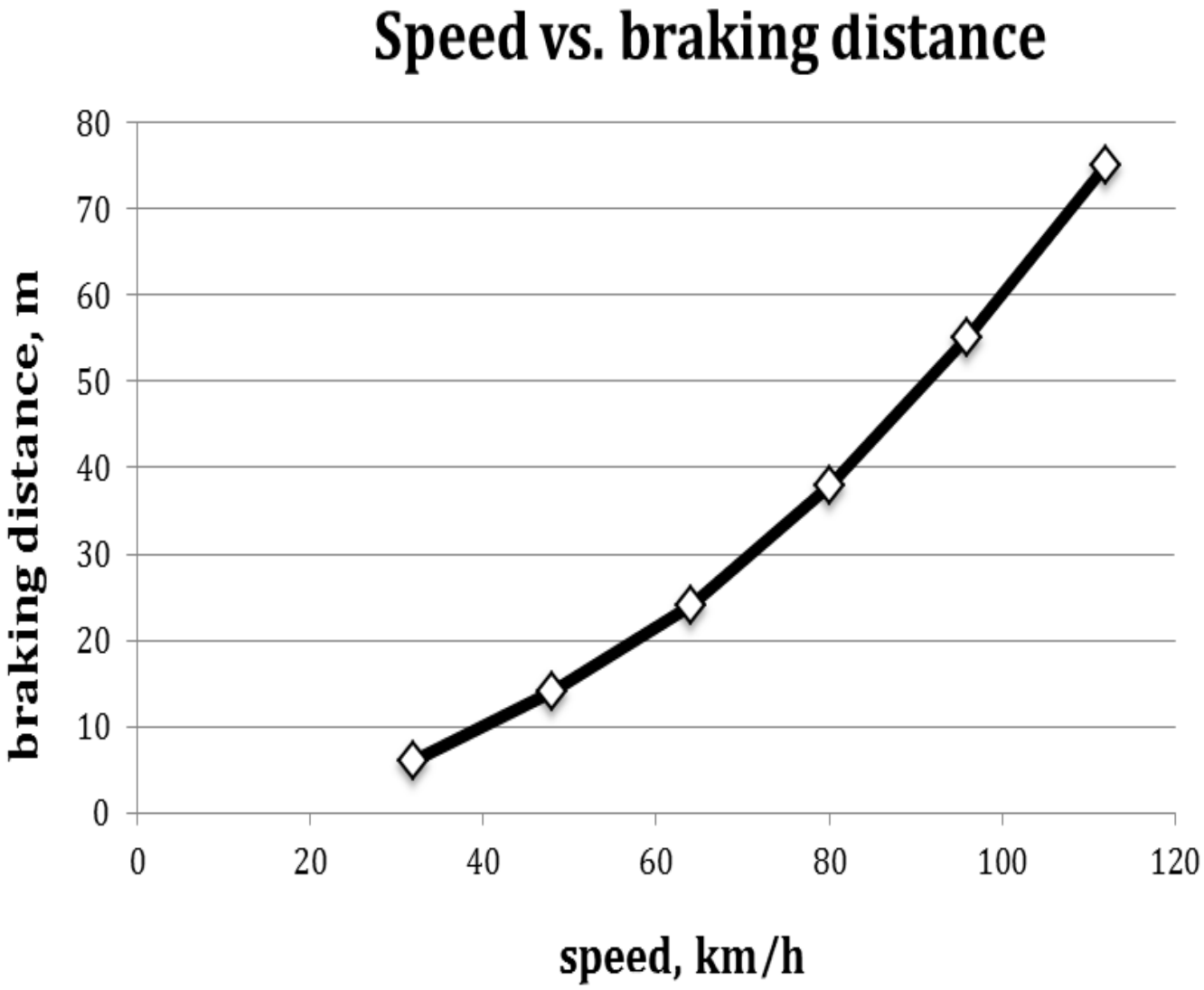


3-1 How ACC work

ACC feature is automated process happened continuously every millisecond, this process contains measure the distance between the car and the front car, then compare it with the safe distance per certain speed, then the car should take fast action to update the car behavior to avoid crushes.

3-2 Safe Distance

The safe distance is the distance required to stop the car in a safe way for its passengers without a crush with the front car or any obstacle in front of the car. Safe distance is not fixed value, it is varying based on the car speed and braking system, that's mean the car moves with speed 20km/h need safe distance less than a car moves with speed 80km/h



3-3 Safe Distance Calculation

In our project we build a tiny scale car so there is difference in materials and mechanical systems, but we have same concepts of ACC which we need to apply it.

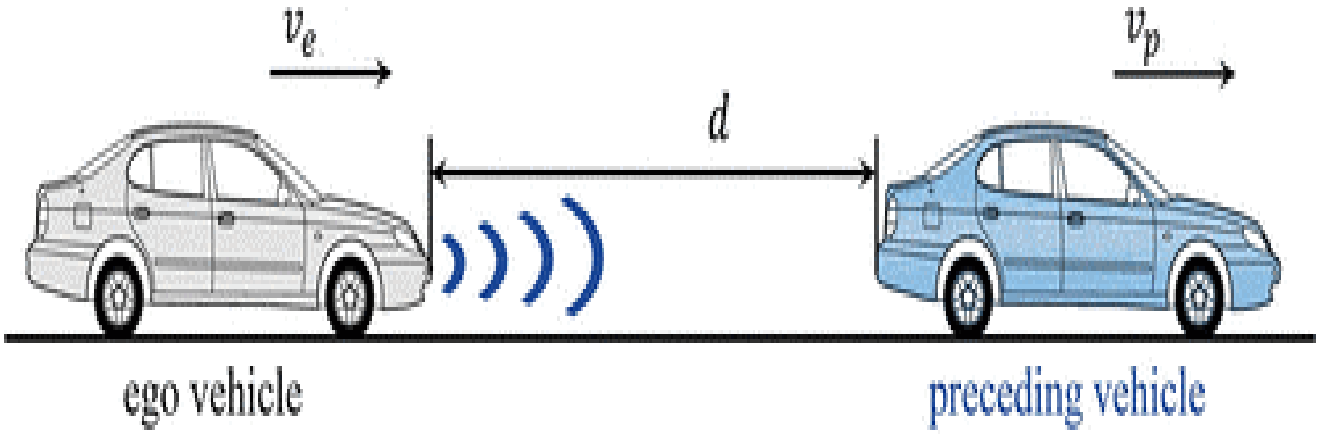
At first, we determine the max speed of the tiny car while giving it max power and all physical loads, after that we calculate the distance that the car moved when it receive order to stop the motors, and by doing many experiments on different speeds we found results as shown in table (2).

N% of Speed	RPM	Cm
20%	135	45
40%	185	62
60%	198	68
80%	210	71
100%	218	74

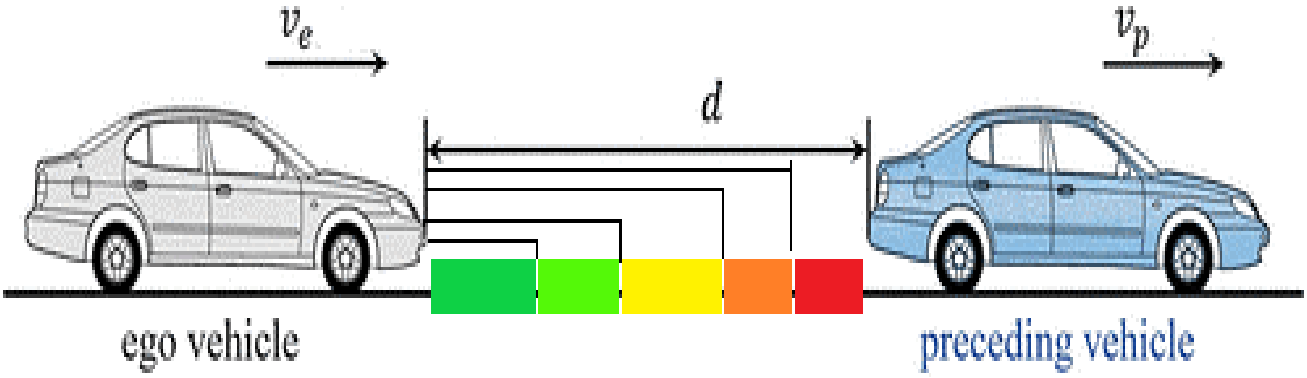
3-4 ACC Mechanism

ACC will provide readings of front distance continuously and ready to take suitable actions to slow down the car, stop it or speed up again when there are no obstacles ahead. Distance calculated by using Ultrasonic Sensor installed at front of the car, the sensor sends waves for certain range with certain speed then receive the waves after time T then calculate the distance **d.**

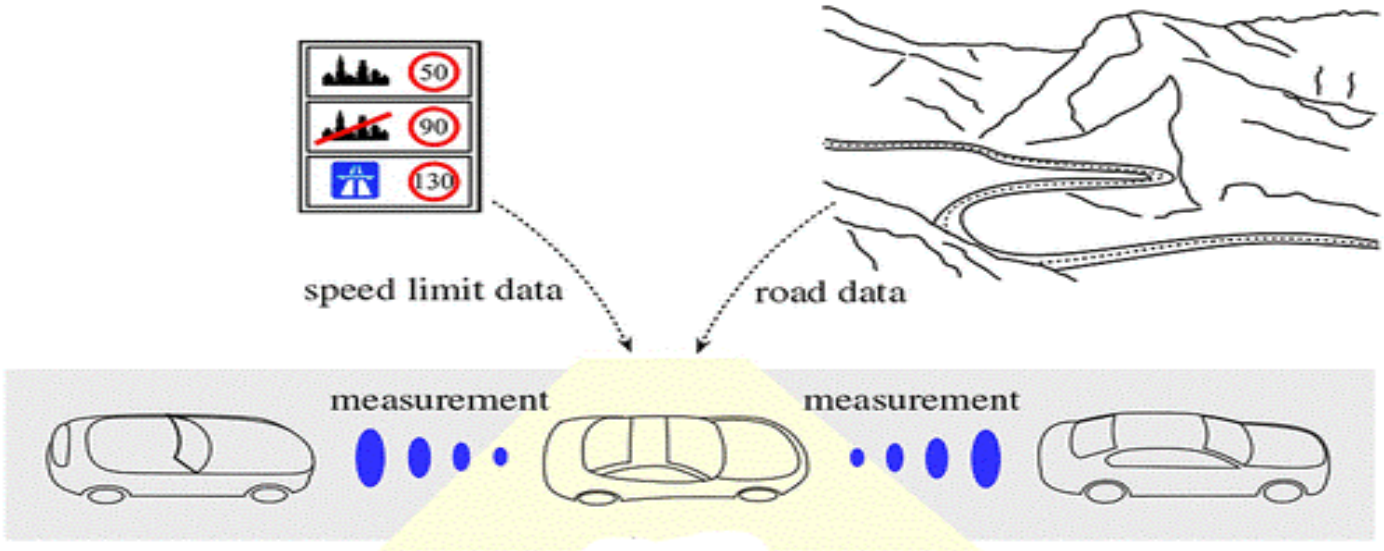
$$d = \frac{wave\ speed}{t/2}$$



After determining the distance between the car and the fronted car, the Microcontroller will decide what action needed based on the distance and the car speed.



In addition to that the car will follow the street signs and GPS instructions during using ACC mode to keep on right track without breaking rules.



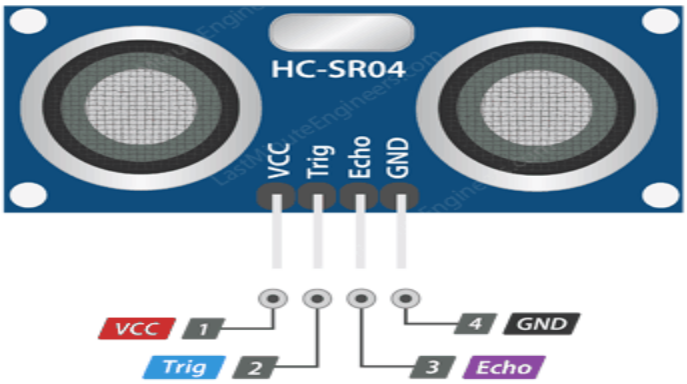
4- Modules and Components

This feature is achieved using

- **Ultrasonic sensor**,
- **2 DC Motors**
- **Motor driver l298N**
- **ATmega32 Microcontroller**.

Hint: maximum range is **4m theoretically** practically **80cm**.

4-1 Ultrasonic Sensor:



So, the actual range of the sensor which we should build on it is 80 cm, we could divide this range to 5 levels each level is equal 20% from the range.

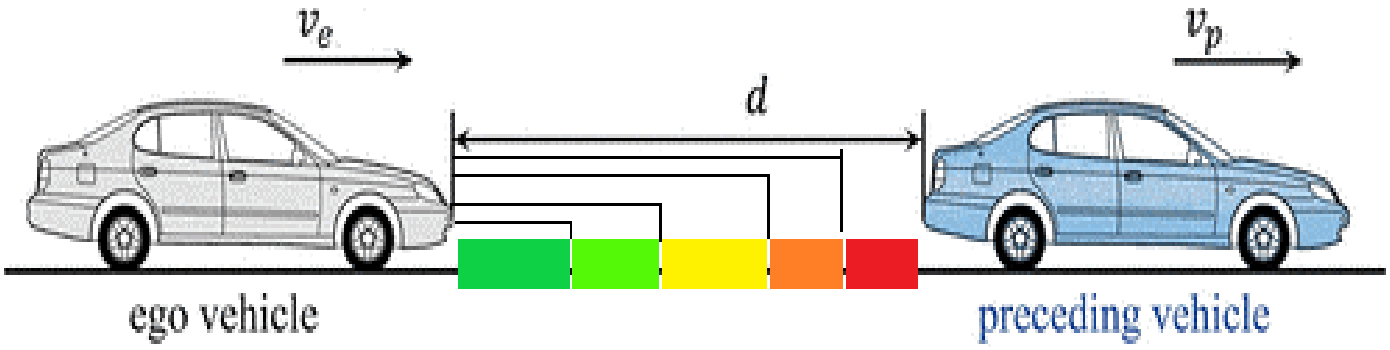
1- Green level: from **80** cm to **64** cm (100% - 80%)

2- Lite Green Level: from **64** cm to **48** cm (80% - 60%)

3- Yellow Level: from **48** cm to **32** cm (60% - 40%)

4- Orange level: from **32** cm to **16** cm (40% - 20%)

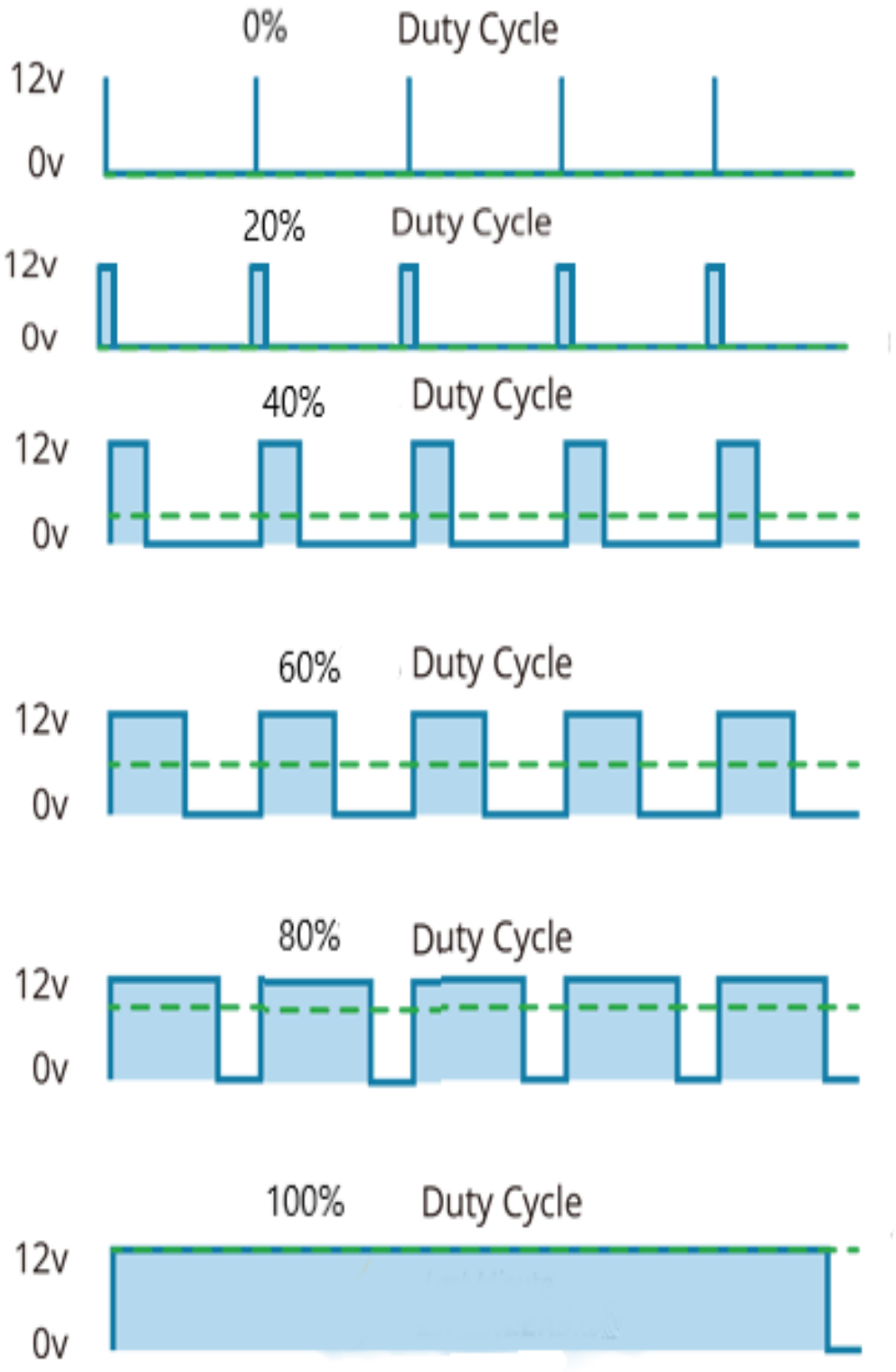
5- Red Level: less than **16** cm (less than 20%)

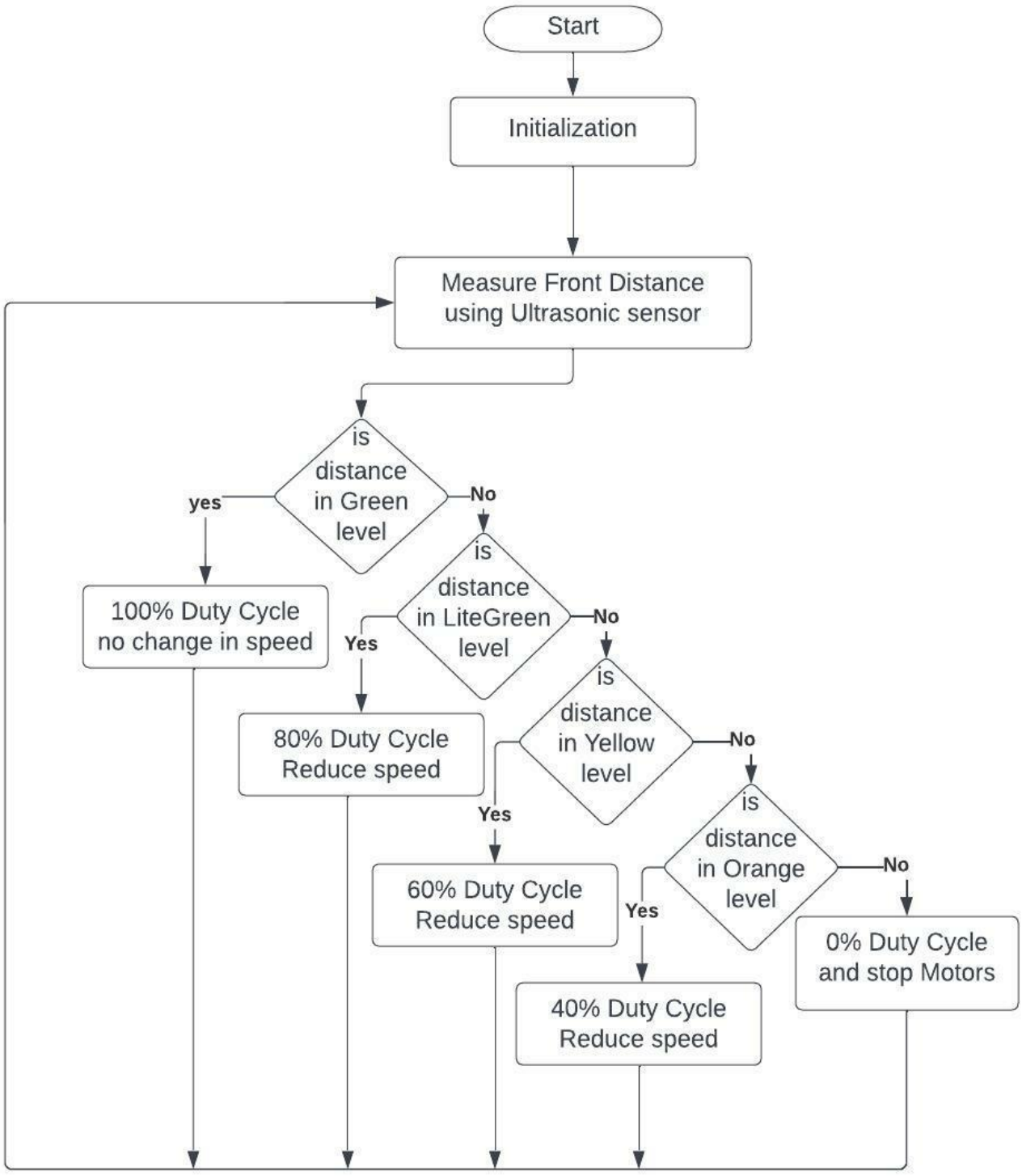


4-2 DC Motors

As mentioned in Self-Parking chapter of controlling a DC Motor part PWM Page 11.

5- ACC Logic Algorithm





As shown in previous flowchart the system will measure at first the front distance then compares it with the levels of distance to determine which decision should be taken.

Every action is about increasing or decreasing car speed by 20% of its speed which will keep passengers safe and will not lead them too sudden braking.

(3) OVERTAKE feature

1. Abstract:

This feature describes the design, implementation, and evaluation the position of a static cars and based then overtaking decision, motion planning, and control algorithm for autonomous vehicles.

Both driver acceptance and safety, when surrounded by other vehicles, must be considered during autonomous overtaking.

These are considered through safe distance based on car driving behaviour.

Since all vehicles cannot be equipped with a vehicle-to-vehicle communications at present, autonomous vehicles should perceive the surrounding environment based on local sensors.

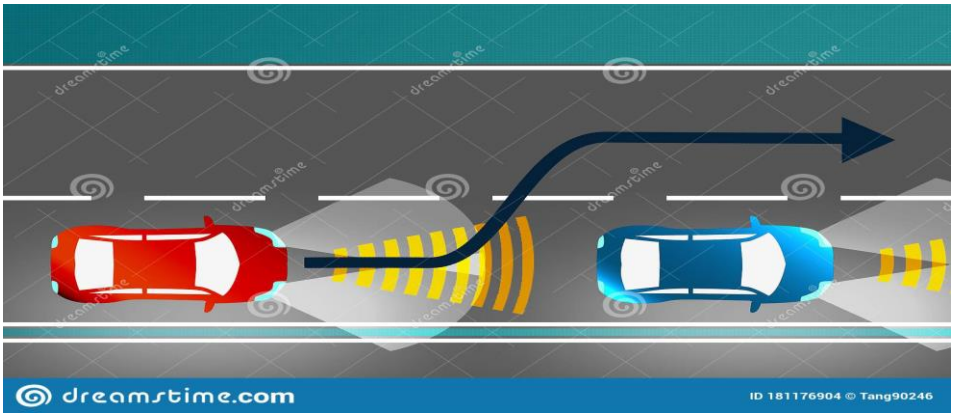
In this feature, static targets are devised to cope with the limitation of cognitive range.

A probabilistic prediction is adopted to enhance safety, given the potential behaviour of surrounding vehicles.

Then, decision-making and motion planning has been designed based on the probabilistic prediction-based safe distance, which could achieve safety performance without a heavy computational burden.

This feature has considered the decision rules that drivers use when overtaking. For this purpose, concepts of target space, demand, and possibility for lane change are devised.

In this paper, three driving modes are developed for active overtaking. The desired driving mode is decided for safe and efficient overtaking. To obtain desired states and constraints, intuitive motion planning is conducted. A stochastic model predictive control has been adopted to determine vehicle control inputs. The proposed autonomous overtaking feature has been evaluated through simulation, which reveals the effectiveness of virtual targets. **[6]**



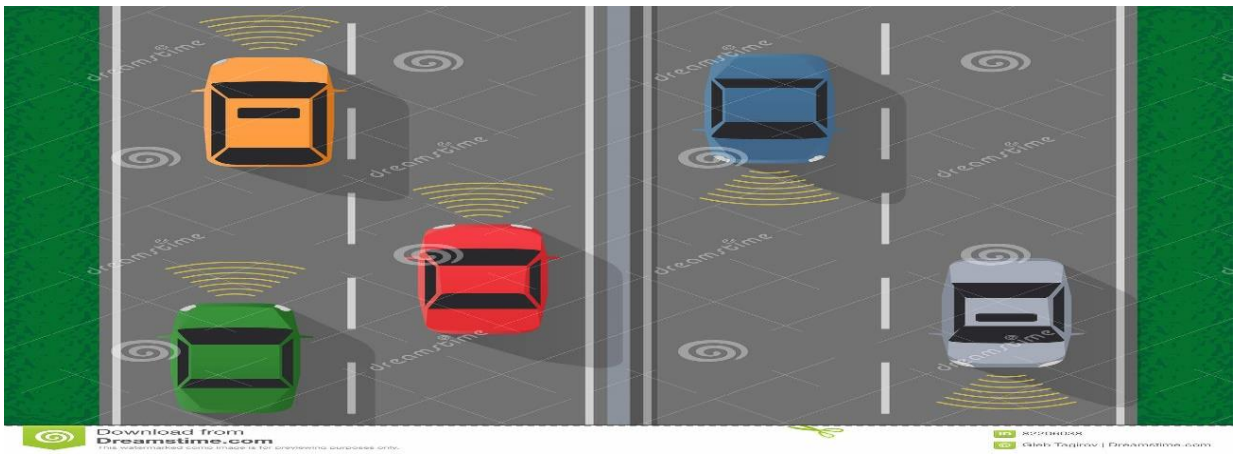
2. Introduction.

The final goal of autonomous vehicle (AV) development is to achieve driverless vehicles that can automatically cope with diverse tasks (e.g., lane keeping, lane change, and overtaking), Among the various tasks, overtaking is a difficult task because of the uncertain behaviour of the surrounding vehicles

An autonomous vehicle is an integrated system consisting of five modules: localization, perception, decision-making, motion planning, and control.

overtaking, when driving, an autonomous vehicle needs to make decisions on diverse driving motions with consideration for the surrounding environment. Decision-making, motion planning, and control modules are important for solving overtaking problems.

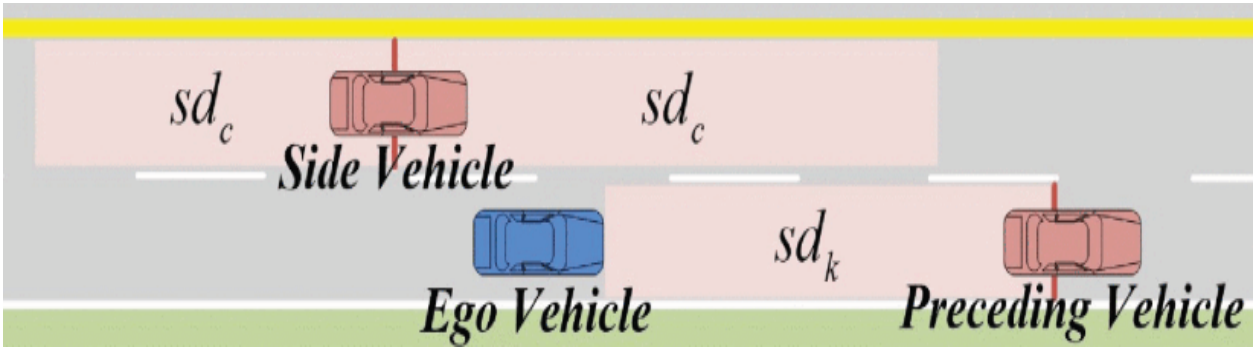
Because all vehicles cannot be equipped with vehicle to vehicle (V2V) communications at present, autonomous vehicles should perceive the surrounding environment based on local sensors. Social perception has been devised to deal with local sensor limits.



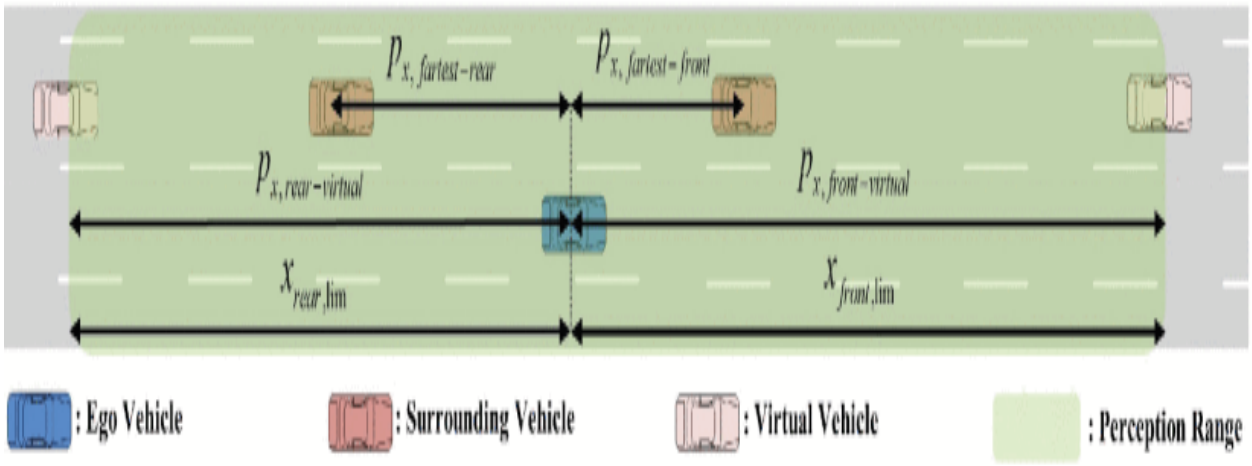
- **Safe Distance:**

The primary purpose of autonomous driving is to drive safely with surrounding vehicles. In this feature, a safe distance concept is adopted as a safety index. In defining safe distance, human driving data and collision avoidance performance are considered. Since overtaking is a complex maneuverer that requires both lane keeping and lane change, different safe distances are used in lane keeping and lane change. In lane keeping situation, preceding and following vehicles could exist. Since safety for preceding vehicle is controllable by the ego vehicle, the preceding vehicle is important. In lane change situation, side vehicles are important. Figure 2 represents safe distances in lane

keeping and lane change situations. The safe distances are derived by analysing the driver data and combining various safety indices.



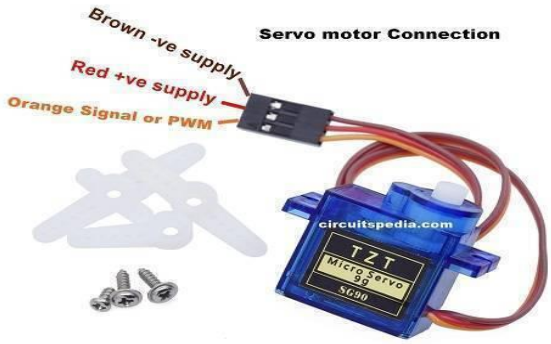
In overtaking situations, interactions with side and behind vehicles are important. When the ego vehicle conducts lane change, side vehicles might suddenly appear from outside of the perception range. In this study, the concept of virtual targets has been developed to cope with the limitation of cognitive range. The concept of virtual targets is shown in Figure **[7]**. Since it is assumed that vehicles always exist at the perception limit, the virtual target can conduct decision-making and motion planning, considering the perception limit.



3. Modules used for overtaking feature:

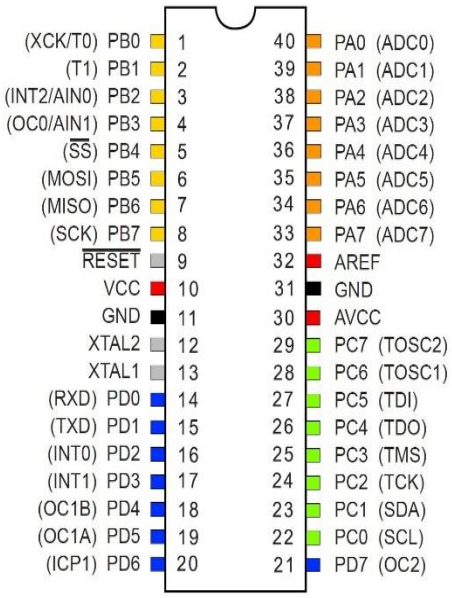
1. **Ultrasonic sensor,** " look at page number 9 "**.**
2. **Motor driver l298N for H- Bridge** " look at page number 12 ".
3. **Servo Motor** " look at page number 15 "**.**
4. **2 or 4 DC Motors**.
5. **ATmega32 Microcontroller**.
6. **USBASP AVR Programmer.**
7. **Jumpers for connection.**

Servo motor connection:

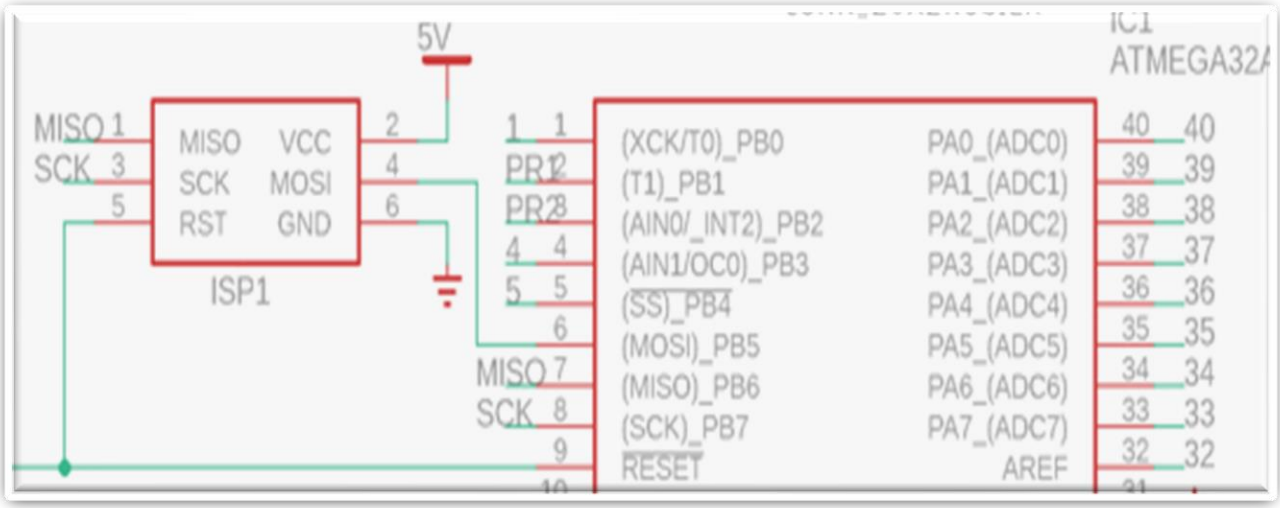


DC Motors:

ATmega32 Microcontroller:



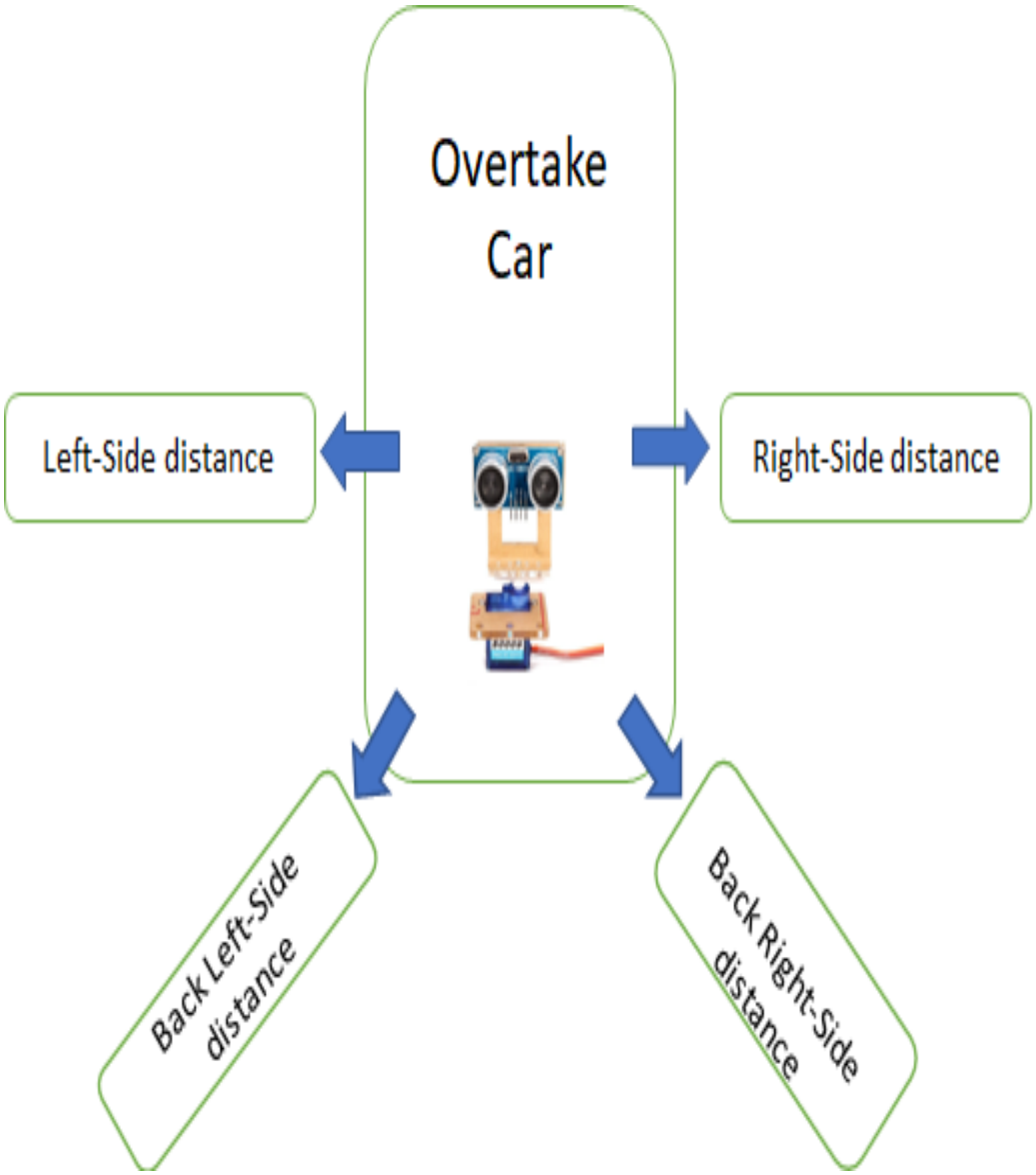
USBASP AVR Programmer:



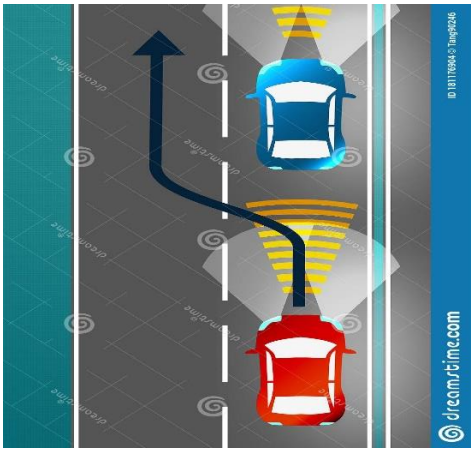
4. Overtake logic algorithm.

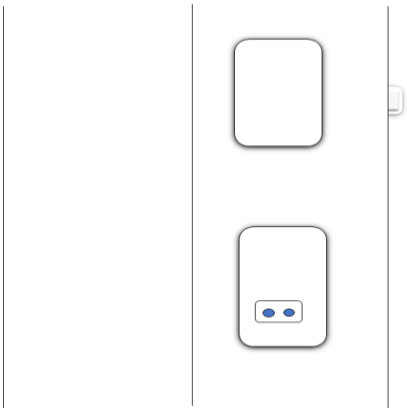
The settings of timer's peripherals will be set as Phase 2 in Self-Parking chapter.

To achieve **Overtake** feature we use ultrasonic sensor installed on servo motor to look like A radar System and positioned as shown:



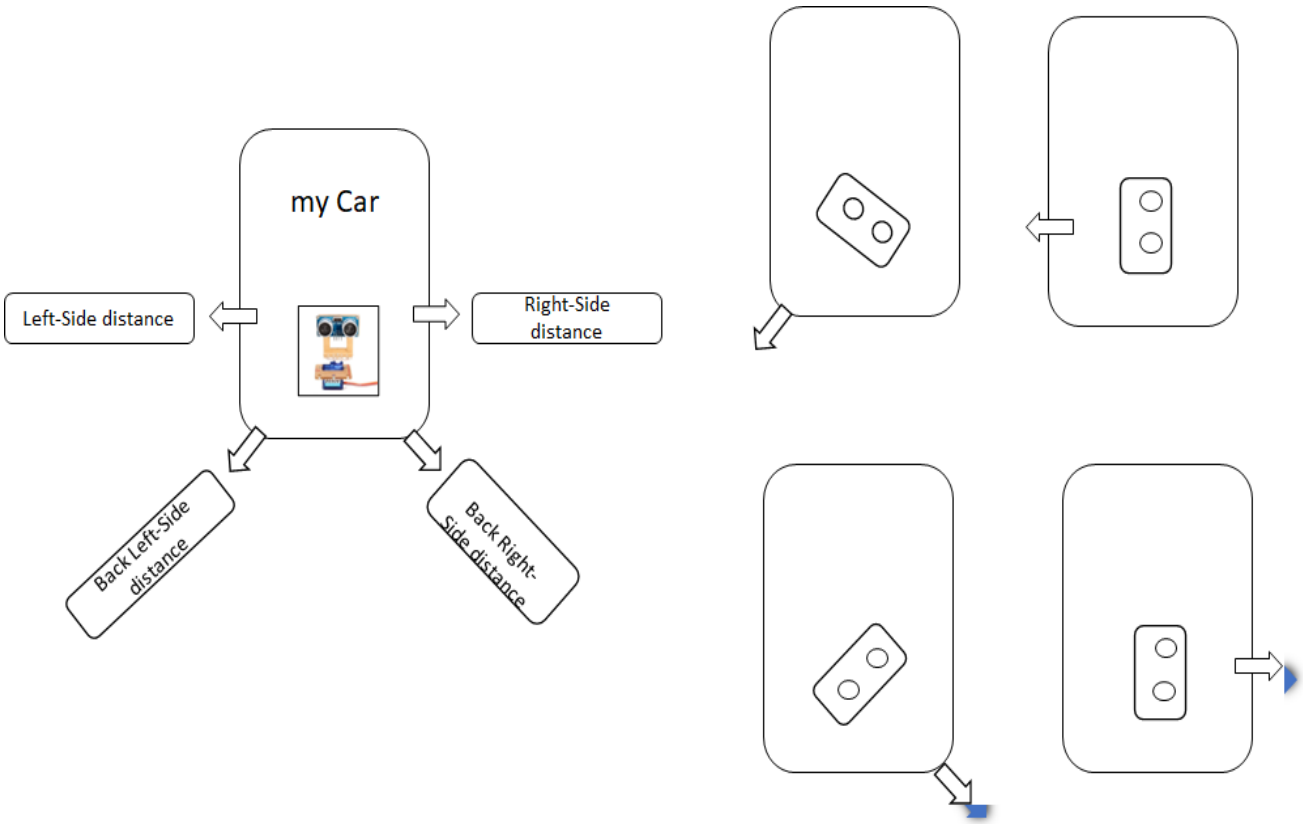
- **Phase 1: There is a stopped car in front of my car.**





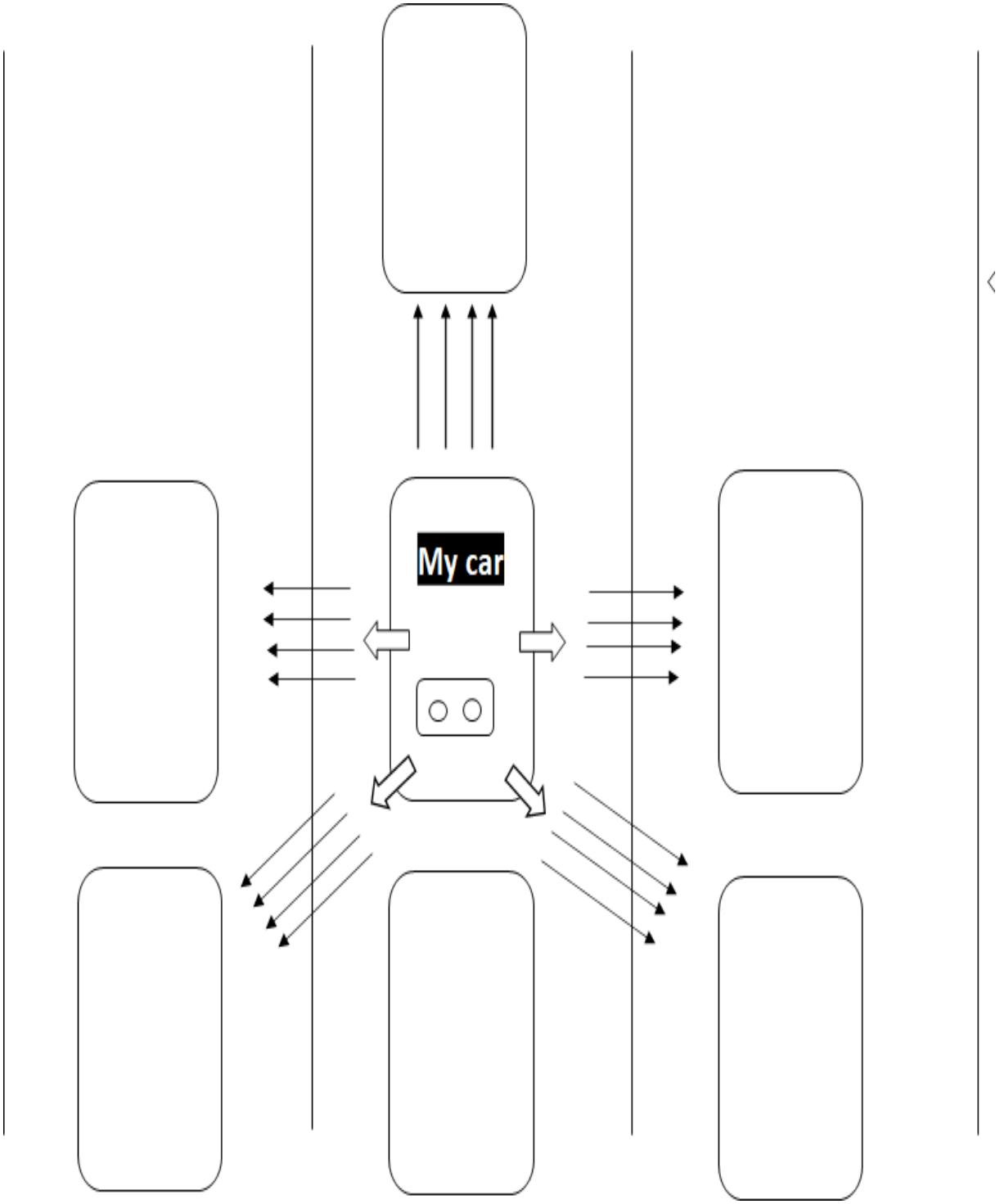
In this case there is a front car was stopped in front of my car with in 10s, in this case the **Radar System** will activate.

The Radar System: consisted of servo motor to move to for 4 angles, angle 0 for left-side distance, angle 45 for Back Left-side distance, angle 135 for Back Right-side distance, angle 180 for Right-Side distance. And at every angle the distance is recorded by Ultra sonic sensor in deferent variables.

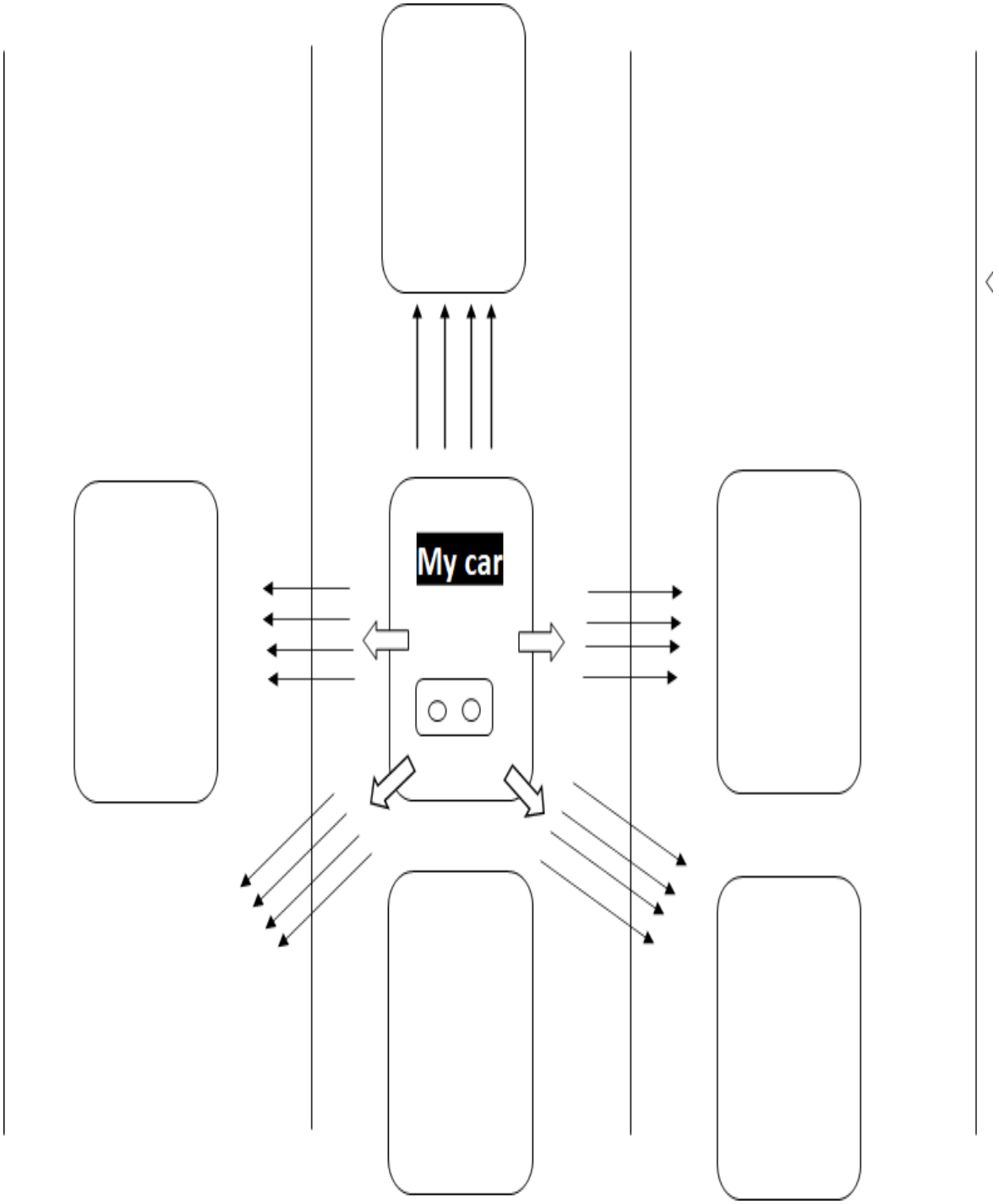


- **Phase 2: Make Decision:**

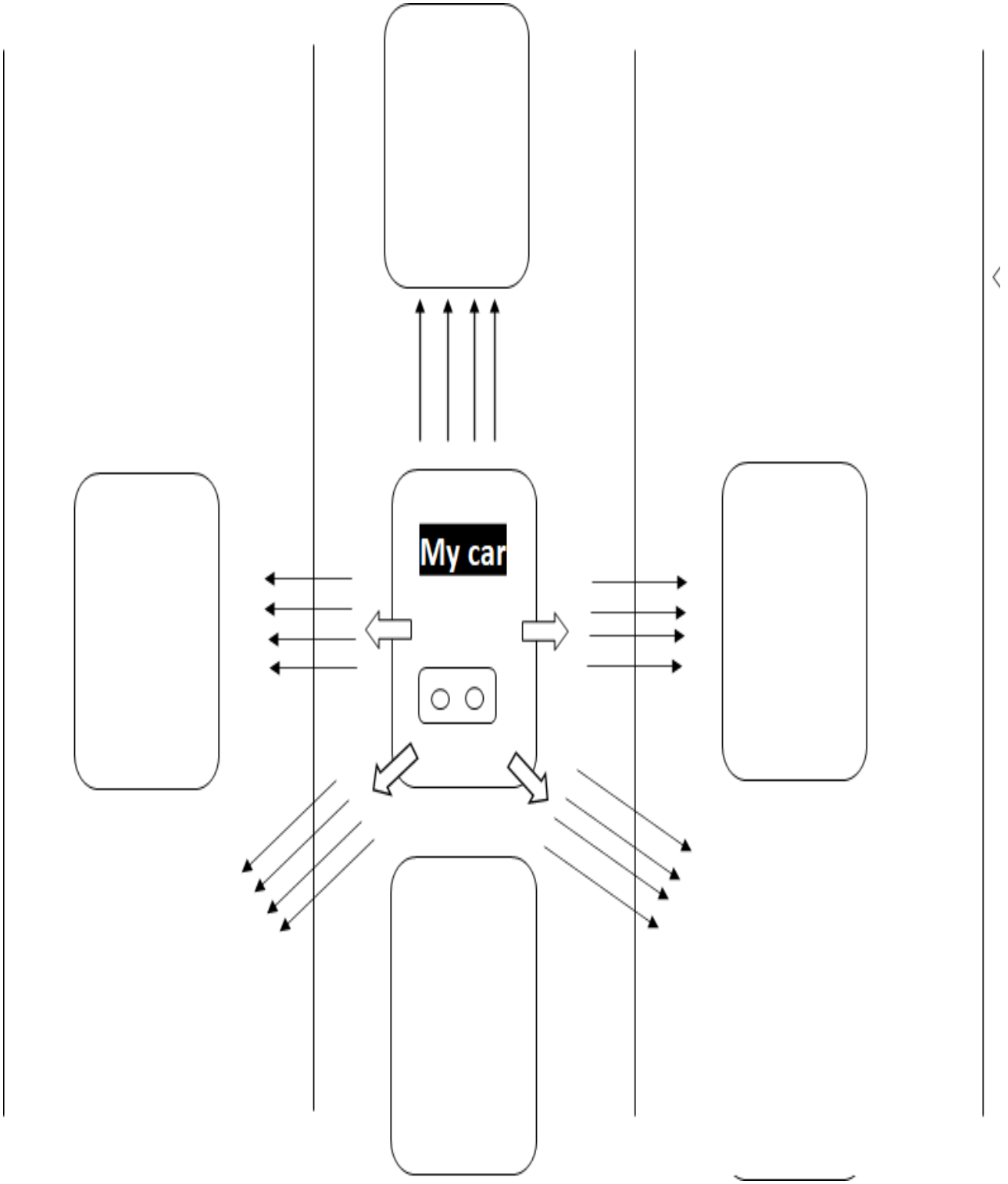
1. If the Radar System read distance in angle 0 for left-side distance < 50cm, angle 45 Back Left-side distance < 65cm, angle 135 for Back Right-side distance < 65cm and angle 180 for Right-Side distance < 50cm, then there is no action will take, because the traffic jam.



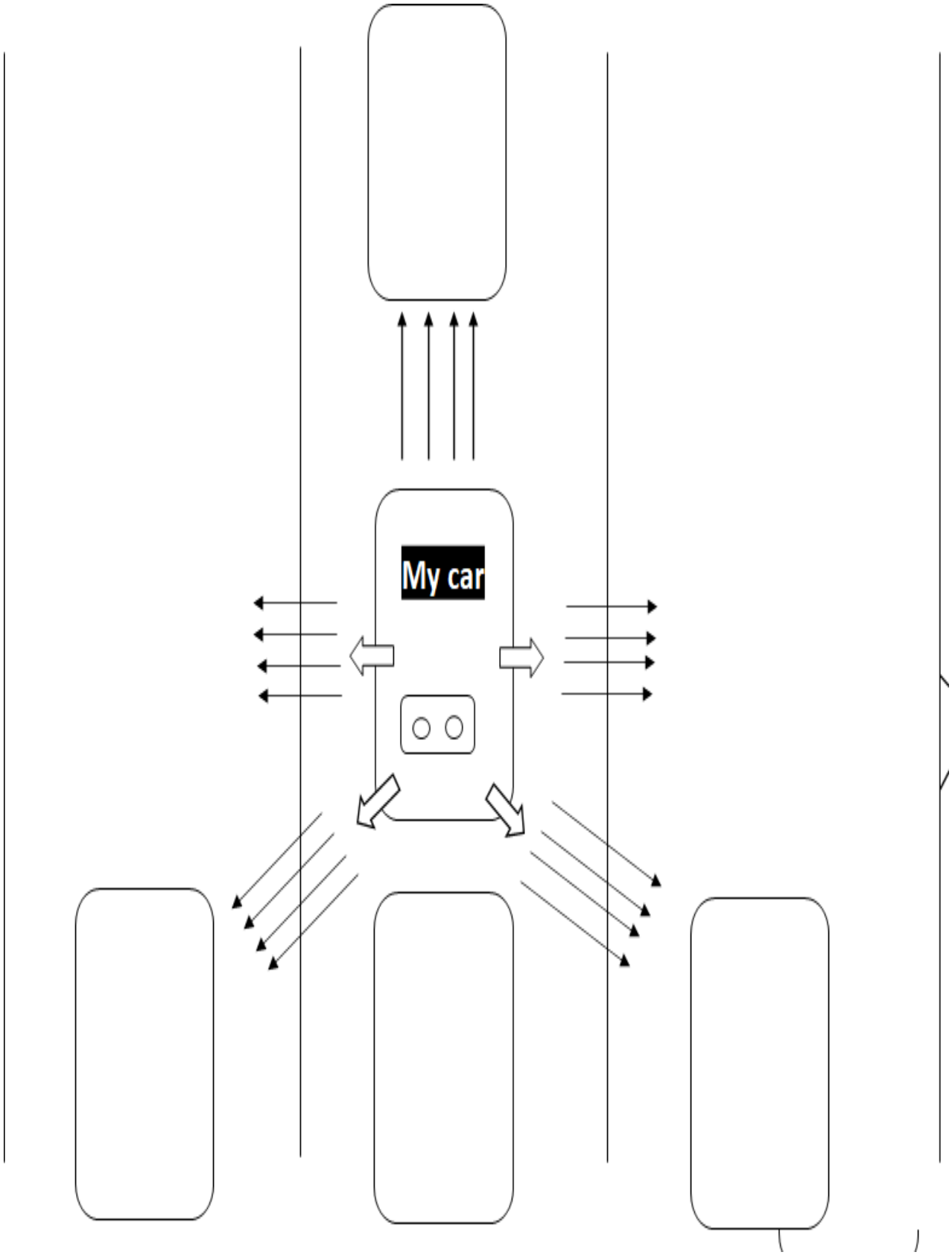
2. If the Radar System read distance in angle 0 for left-side distance < 50cm, angle 45 Back Left-side distance > 65cm, angle 135 for Back Right-side distance < 65cm and angle 180 for Right-Side distance < 50cm, then there is no action will take, because the traffic jam and the side left car.



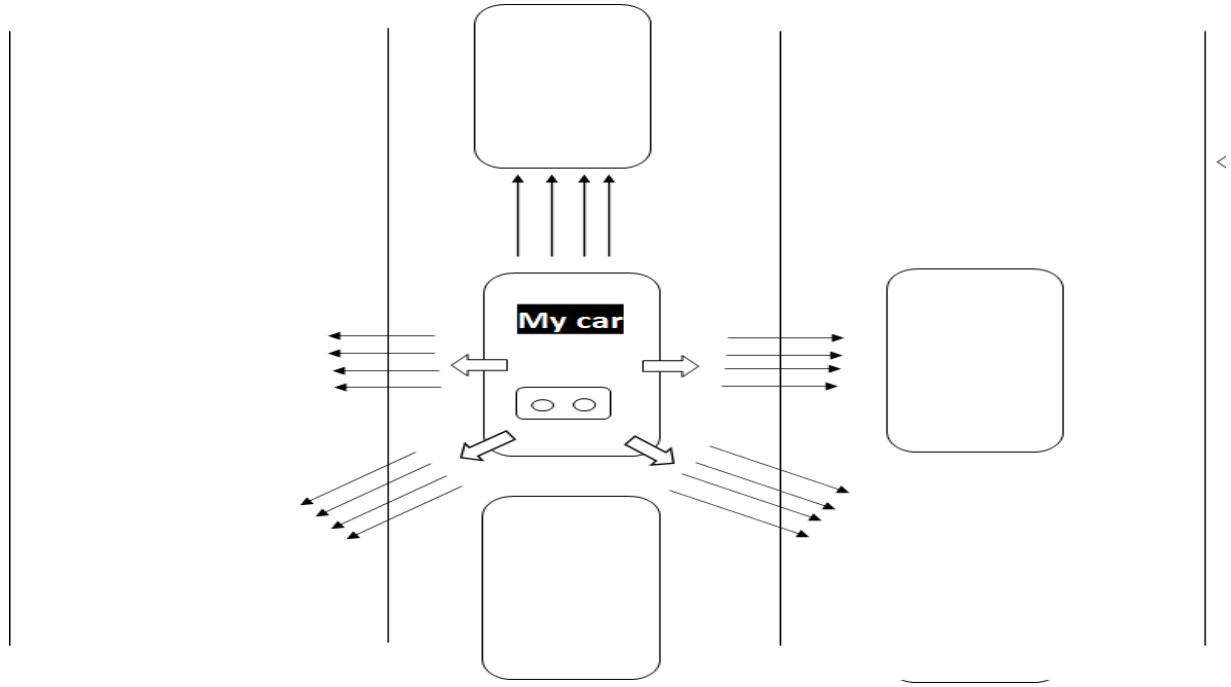
3. If the Radar System read distance in angle 0 for left-side distance < 50cm, angle 45 Back Left-side distance > 65cm, angle 135 for Back Right-side distance > 65cm and angle 180 for Right-Side distance < 50cm, then there is no action will take, because the traffic jam and the side Left & Right car.

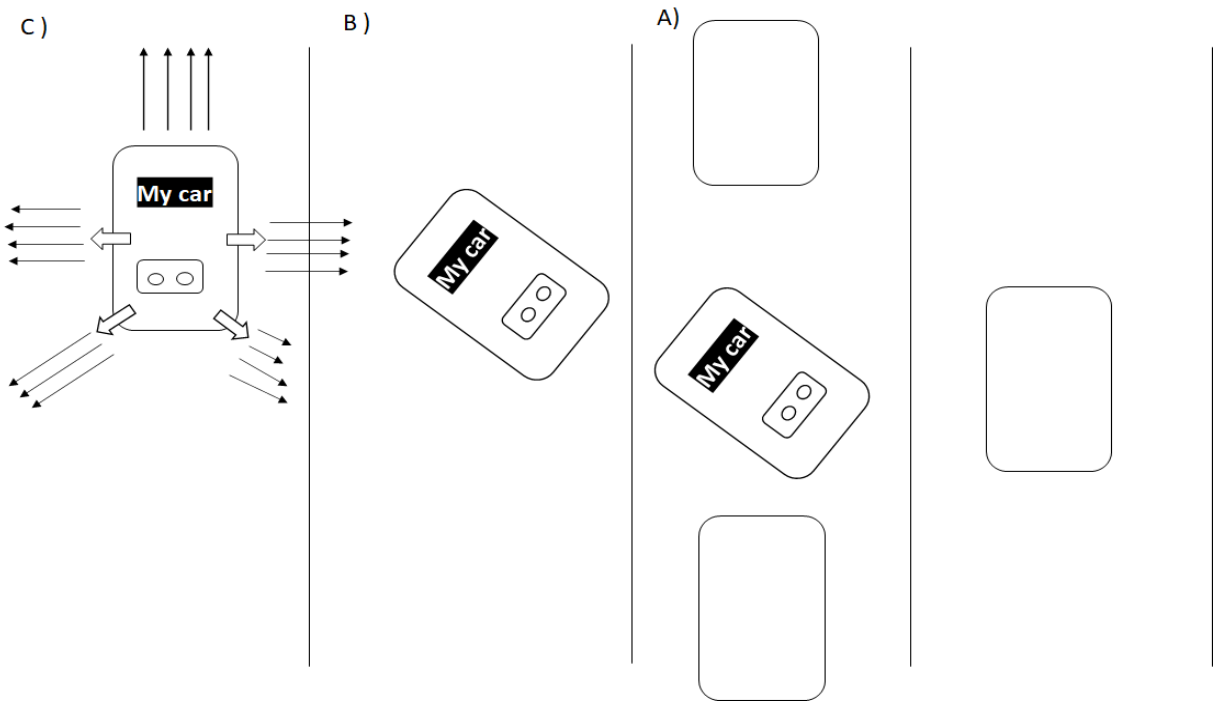


4. If the Radar System read distance in angle 0 for left-side distance > 50cm, angle 45 Back Left-side distance < 65cm, angle 135 for Back Right-side distance < 65cm and angle 180 for Right-Side distance >50cm, then there is no action will take, because the traffic jam and the Back Right-side Left & 45 Back Left-side cars.

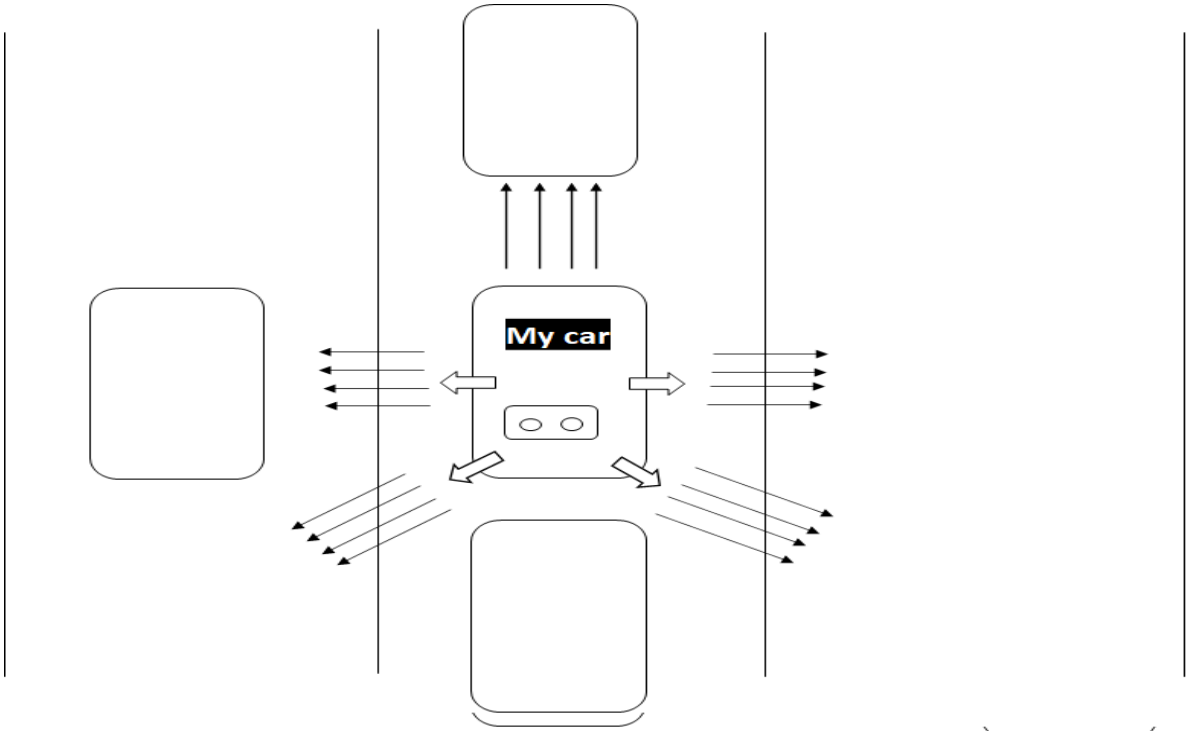


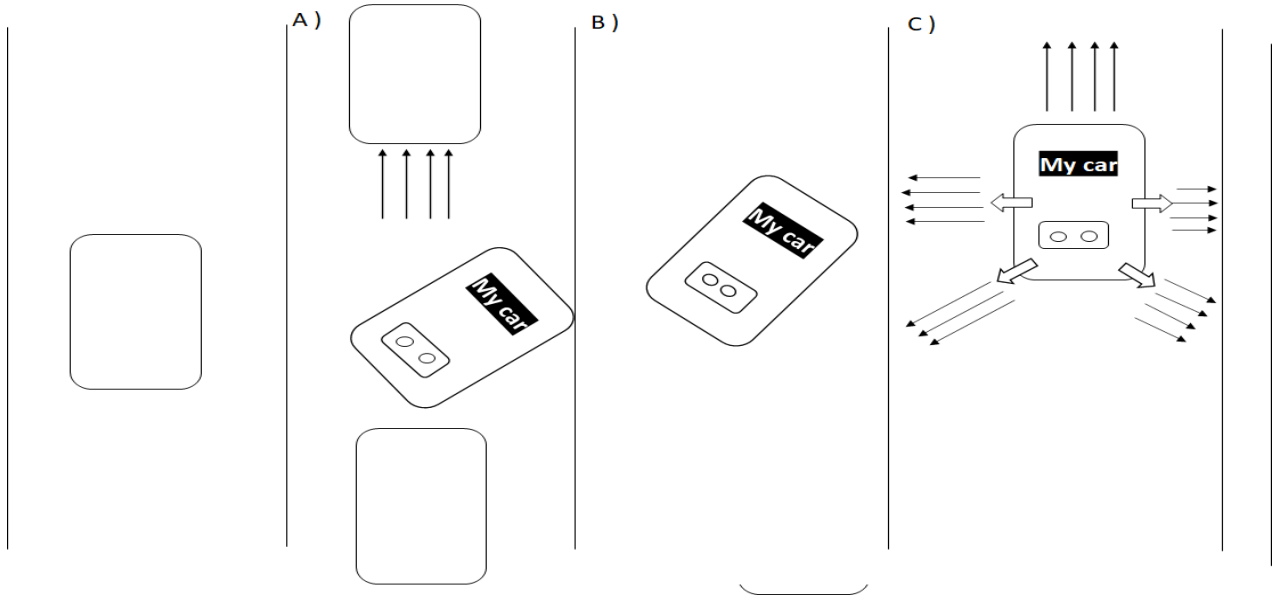
5. If the Radar System read distance in angle 0 for left-side distance > 50cm , angle 45 Back Left-side distance > 65cm ,angle 135 for Back Right-side distance > OR < 65cm and angle 180 for Right-Side distance < 50cm, then the action will take is **moving Left function** , the car will rotate left with angle 45 degree with delay 500ms then move forward for delay 1000s then rotate right with angle 0 with delay 500ms , move forward and finally call ACC(); .



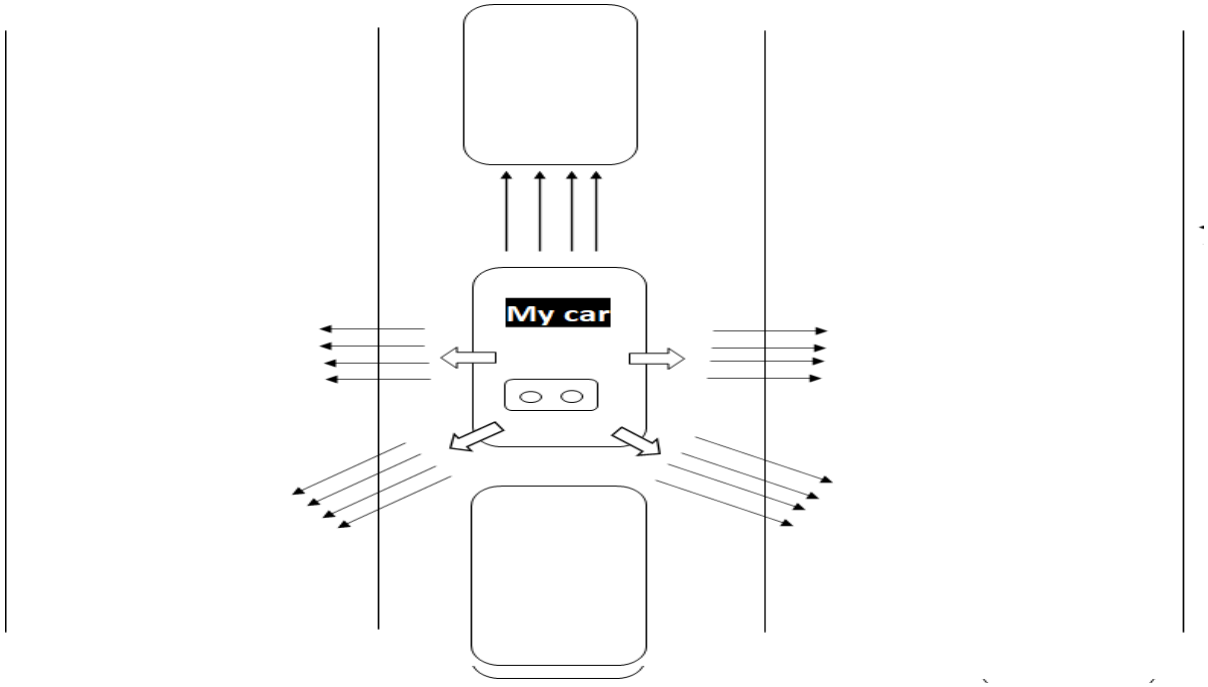


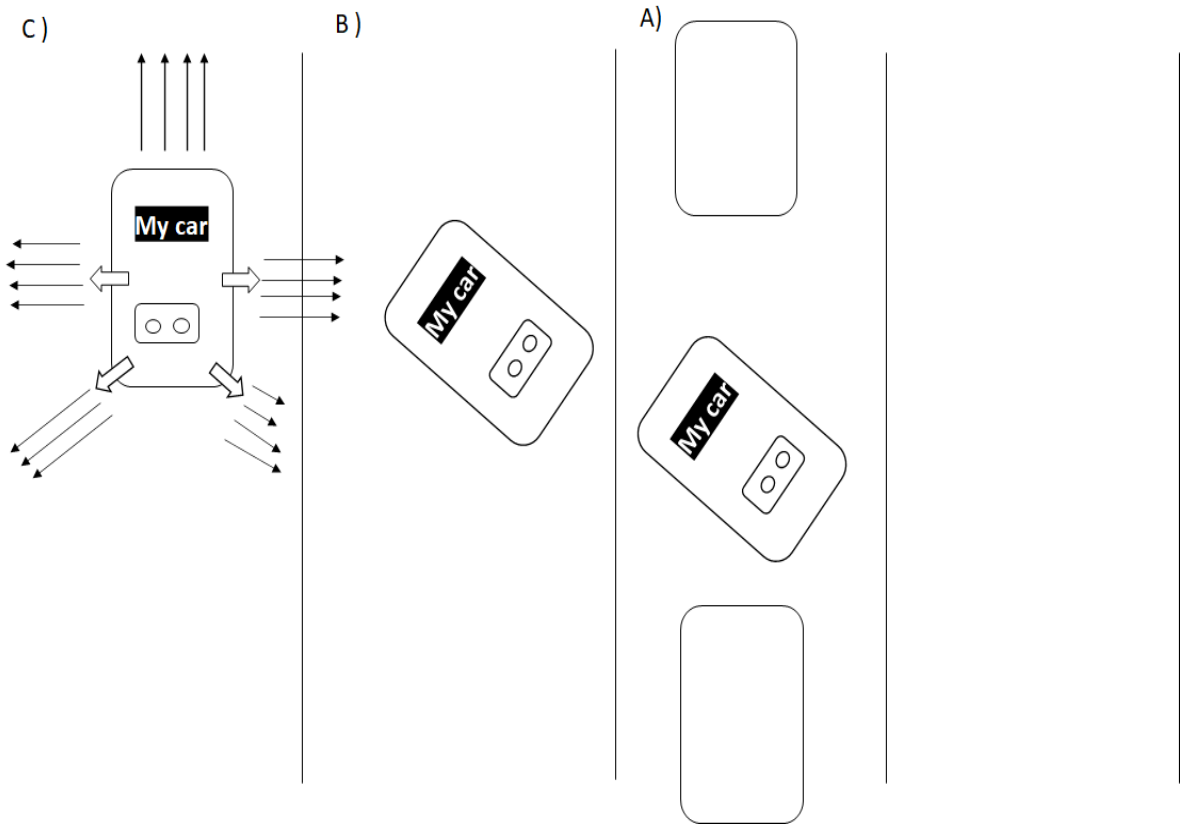
6. If the Radar System read distance in angle 0 for left-side distance < 50cm , angle 45 Back Left-side distance > **OR** < 65cm ,angle 135 for Back Right-side distance > 65cm and angle 180 for Right-Side distance > 50cm, then the action will take is **moving Right function** , the car will rotate Right with angle 45 degree with delay 500ms then move forward for delay 1000s then rotate Left with angle -45 degree with delay , move forward with angle 0 and finally call ACC(); ..





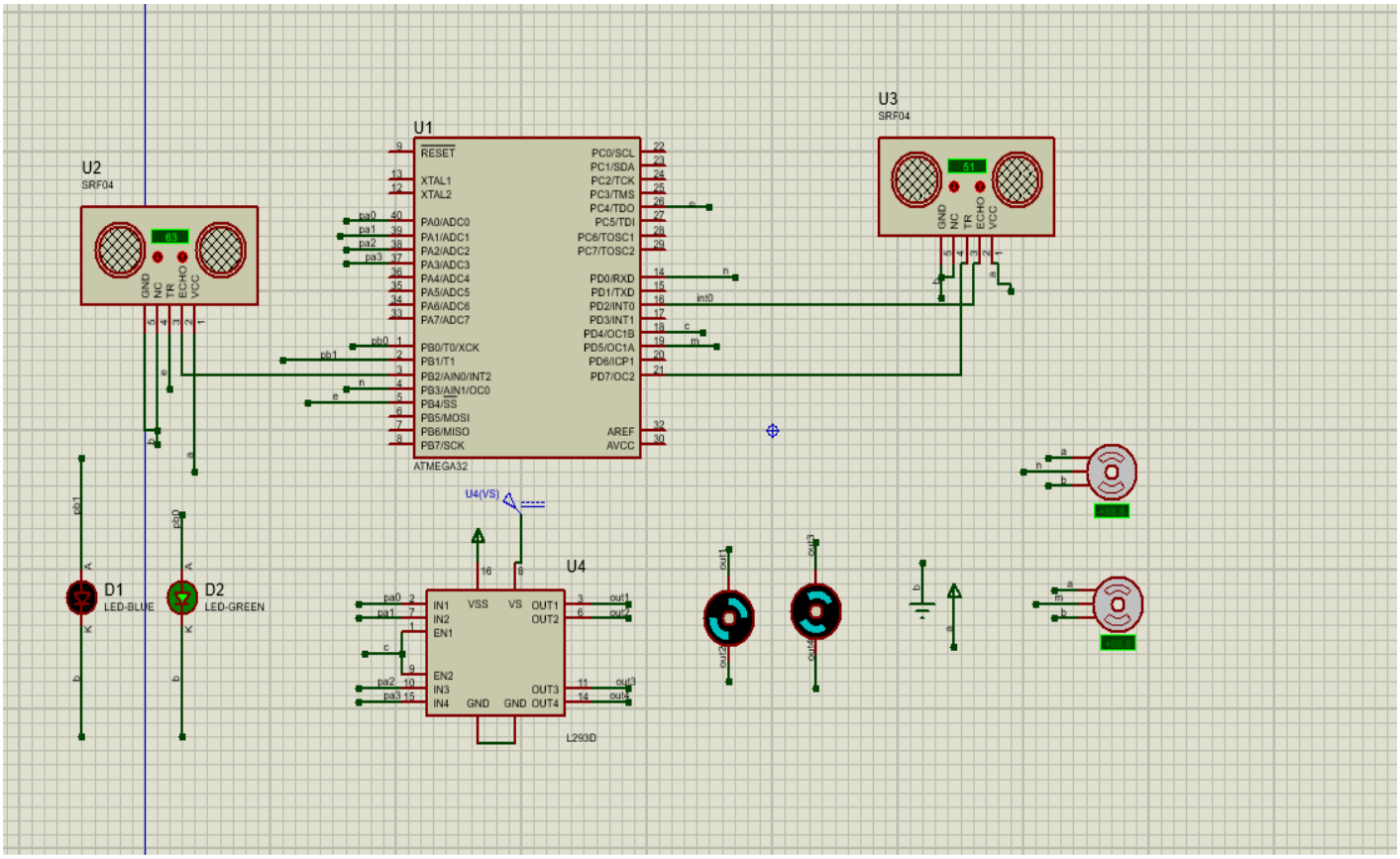
7. If the Radar System read distance in angle 0 for left-side distance > 50cm , angle 45 Back Left-side distance > 65cm ,angle 135 for Back Right-side distance > 65cm and angle 180 for Right-Side distance > 50cm, then **"any direction"** ,the action will take is **moving Left function** , the car will rotate left with angle 45 degree with delay 500ms then move forward for delay 1000s then rotate Right with angle -45 degree with delay 500ms ,move forward with angle 0 and finally call ACC(); .





Left-side distance	Back Left-side distance	Back Right-side distance	Right-side distance	ACTION
< 50cm	< 65cm	< 65cm	< 50cm	No Action and current position
< 50cm	> 65cm #Not care	< 65cm	< 50cm	No Action and current position
< 50cm	> 65cm #Not care	> 65cm #Not care	< 50cm	No Action and current position
> 50cm	< 65cm	< 65cm	> 50cm	No Action and current position
> 50cm	> 65cm	< 65cm	< 50cm	Action Move Left function
< 50cm	< 65cm	> 65cm	> 50cm	Action Move Right function
> 50cm	> 65cm	> 65cm	> 50cm	Action any direction Move Left function

Simulation OVERTAKE feature:



(4) Traffic Sign Recognition

1- Introduction:

Traffic signs are indicators present at either the side or above roads to provide information about the road conditions and directions. Traffic signs are generally pictorial signs using symbols and pictures to give information to the vehicle drivers. There are various categories of traffic signs classified according to the information they convey**.**

There are primarily 7 different traffic signs:

1) Warning signs, 2) Priority signs, 3) Restrictive signs, 4) Mandatory signs,

5) Special regulation signs, 6) Informative signs, 7) Directional signs and additional panels.

Among these signs, warning signs are most important. Warning signs indicate the potential hazard, obstacle, condition of the road. The warning signs may provide information about the state of the road or hazards on the road which the driver may not immediately see. This category includes pedestrian crossing sign, caution sign, crosswalk alert, traffic light alert etc. Priority signs indicate the course of the vehicle, i.e., the way a vehicle should pass to prevent a collision. This category includes stopping sign, intersection sign, one-way traffic sign etc. Restrictive signs are used to restrict certain types of maneuvers and to prohibit certain vehicles on roads. This includes a no-entry sign, no-motorcycle sign, no-pedestrian etc. Mandatory signs are quite the opposite to prohibit signs saying the drivers what they must do. Permitted directions, Vehicle allowances belong to this category. Special regulation and informative signs indicate regulation or provide information about using something. One way sign, home zone indication comes to this category. Directional signs provide information about the possible destinations from a location. Turn-right, Turn-left etc., come under this category.

In recent years, several attempts are being made to detect the traffic signs and inform the vehicle driver. Traffic Sign Recognition regulates the traffic signs and informs the vehicle driver about the detected sign for efficient navigation, ensuring safety. A smart real-time automatic traffic sign detection can support the driver or, in the case of self-driving cars, can provide efficient navigation. Automatic traffic sign recognition has practical

applications in Driver Assistance System. DAS involves automatic traffic sign recognition where the traffic sign is detected and classified in real-time. Traffic sign recognition is also an important module in unmanned driving technology.

Traffic sign recognition systems usually have two phases. The first phase involves detecting the traffic sign in a video sequence or image using a method called image processing. The second phase involves classifying this detected image using an artificial neural network model. Detecting the traffic signs at various places and recognizing them has been quite an issue nowadays, which causes various problems. According to World Health Organization (WHO), enforcing speed limits on traffic help minimizing road accidents. This speed limit is often neglected by drivers who drive past the limits. Road accidents can be caused due to various reasons like the low quality of the traffic sign, driver negligence, obstacles near the traffic sign, which causes visual hindrance. The mentioned causes involve human-caused errors. Automated traffic sign detection may minimize human-involved errors and can effectively detect the traffic signs.

2- Aim and Objectives:

This thesis aims to build a cost-sensitive CNN model trained and tested using the GTSRB dataset. Later the model is used to classify the images placed in front of a camera detected using OpenCV. **[8]**

2.1 Objectives:

- The first objective is to split the dataset, which contains images, into train and test data.
- To pre-process the images in the dataset using grayscale and histogram equalization methods.
- To build a cost-sensitive CNN model with a modified LeNet model.
- To train and test the model with the corresponding datasets.
- To test the hypothesis of whether the accuracy increases with parameter tuning and draw a suitable conclusion.
- To test the model with the images detected using OpenCV.

3- Motivation:

The built model must be tested in real-time. We send an image extracted from the live camera using OpenCV to input the model for efficient classification. Data pre-processing needs to be done to recognize the image effectively. Grayscale and histogram equalization techniques are suitable for the given dataset.

4- Related Work:

Previously various classifiers and object-classification algorithms were used for the recognition of the traffic signs. Support vector machine, random forest, neural network, deep learning models, decision fusion. Among these algorithms, deep learning models have proven effective in many fields, including speech recognition, Natural Language Processing (NLP), and Image classification. There are many deep learning classifiers like Convolution Neural Network (CNN), Deep Boltzmann Machines (DBM) for image classification.

Zhongqin et al had performed Traffic sign classification using improved VGG-16 architecture and attained 99% model accuracy with the German Traffic Sign Recognition Benchmark (GTSRB) dataset. In their proposed model, redundant layers are removed from the VGG-16 architecture. Also, batch normalization and pooling layers were added to greatly reduce the parameters and accelerate the model's accuracy.

Shangzheng proposed a fast traffic sign recognition method based on an improved CNN model by image processing and machine vision processing technology along with in-depth learning in target classification. The proposal uses HOG features to detect the traffic signs and an improved CNN model to determine the traffic sign. The proposed model accurately locates the traffic sign with improved accuracy and reduced false detection rate.

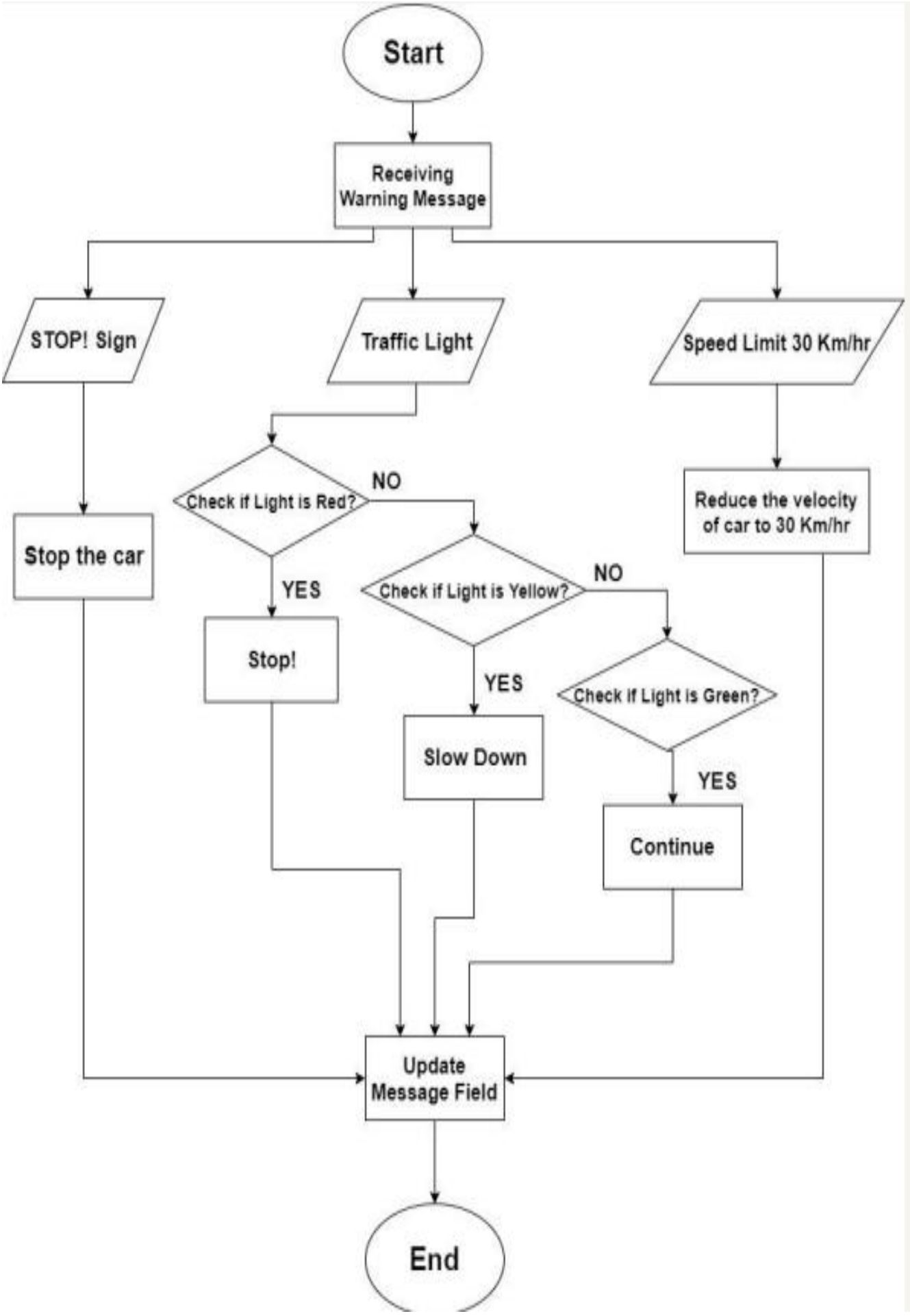
Tsoi and Wheelus had performed TSR with cost-sensitive deep learning CNN model and attained 86.5% model accuracy. Their study built a model to classify traffic signals with cost-sensitive techniques like cost-sensitive rejection sampling and the use of the cost-sensitive loss functions. Their baseline model consists of 3 convolution layers, 2 max-pooling layers, 3 fully connected layers, 2 dropout layers and a SoftMax layer. Shabarinath and Muralidhar proposes an optimized CNN architecture based on VGGNet along with image processing techniques for traffic sign classification. They have utilized the German Traffic Sign Detection Benchmark (GTSDB) for testing and training their model. With Google's TensorFlow running in the background, they achieved 99% validation accuracy and 100% test accuracy for novel input inference. The authors applied pruning and post-training quantization for the trained model to obtain less memory footprint of the CNN.

Dhar et al had performed TSR in real-time with image acquisition techniques using color ques from the HSV color model. The desired area in the image is cropped using region properties and shape signature. The extracted image is thus classified using a CNN model with two convolutional layers. The authors have achieved 97% using Bangladesh Traffic signs where the extracted image is classified using CNN. The authors have also performed the classification using various other classifiers such as SVM (cubic and quadratic), ANN, Decision trees, KNN, Ensembles and achieved the highest accuracy for the CNN classifier. Islam has used LeNET architecture for the classification of the traffic signs. The author apprises two models based on CNN, one for the sign classification while the other is for shape classification. This model was not built for real-time implementation but can be helpful. Shopa, Sumitha and Patra presented a study to recognize traffic sign pattern using OpenCV techniques. In their work, they have used the HSV algorithm for image detection and Gaussian filter for image extraction. Image recognition is done by implementing KNN methods. Image pre-processing, feature extraction and classification are the three modules used to recognize the traffic sign. In the above-mentioned works, either the authors use CNN for traffic sign recognition or OpenCV for traffic sign detection and extraction. Our paper proposes a CNN model to classify the traffic sign and test this model using the images extracted from real-time using OpenCV. The proposed CNN model is based on the LeNET model but has 4 convolution layers, 2 pooling layers, 2 dropout layers, followed by an output layer.

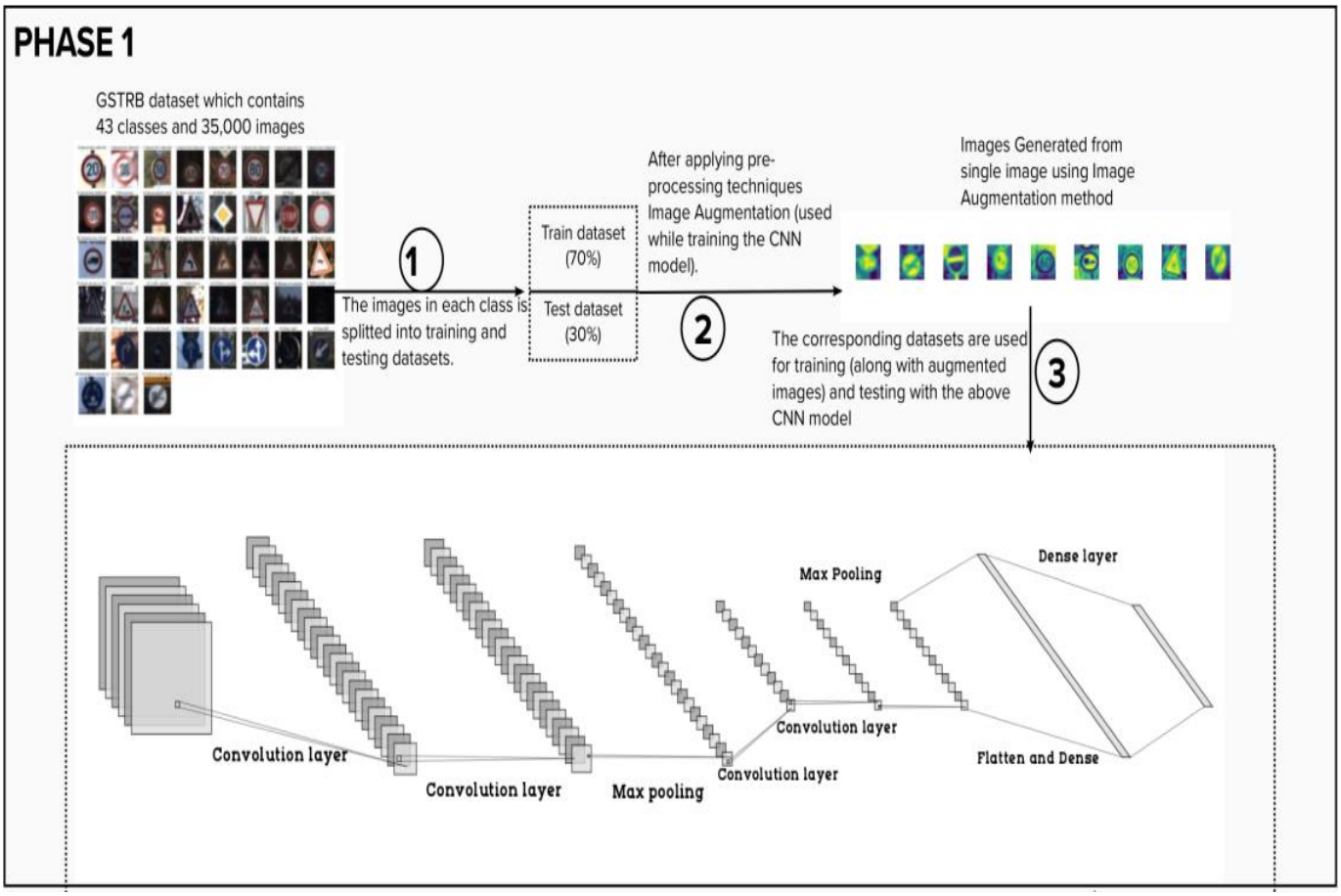
5- Method:

To achieve the aim of the thesis and to address the research questions, we employed the experimentation method in our thesis. A CNN model based on the modified LeNET model was built to conduct the experiment. GTSRB dataset is used to train and test the model to achieve good accuracy. An experiment is done by running the model after parameter tuning to find if it makes a significant change in the model's accuracy **[9]**. Later, another experiment is done where the built model is used to recognize and classify the traffic sign image extracted using OpenCV methods in real-time **[10]**. The procedure of this experimentation is as follows:

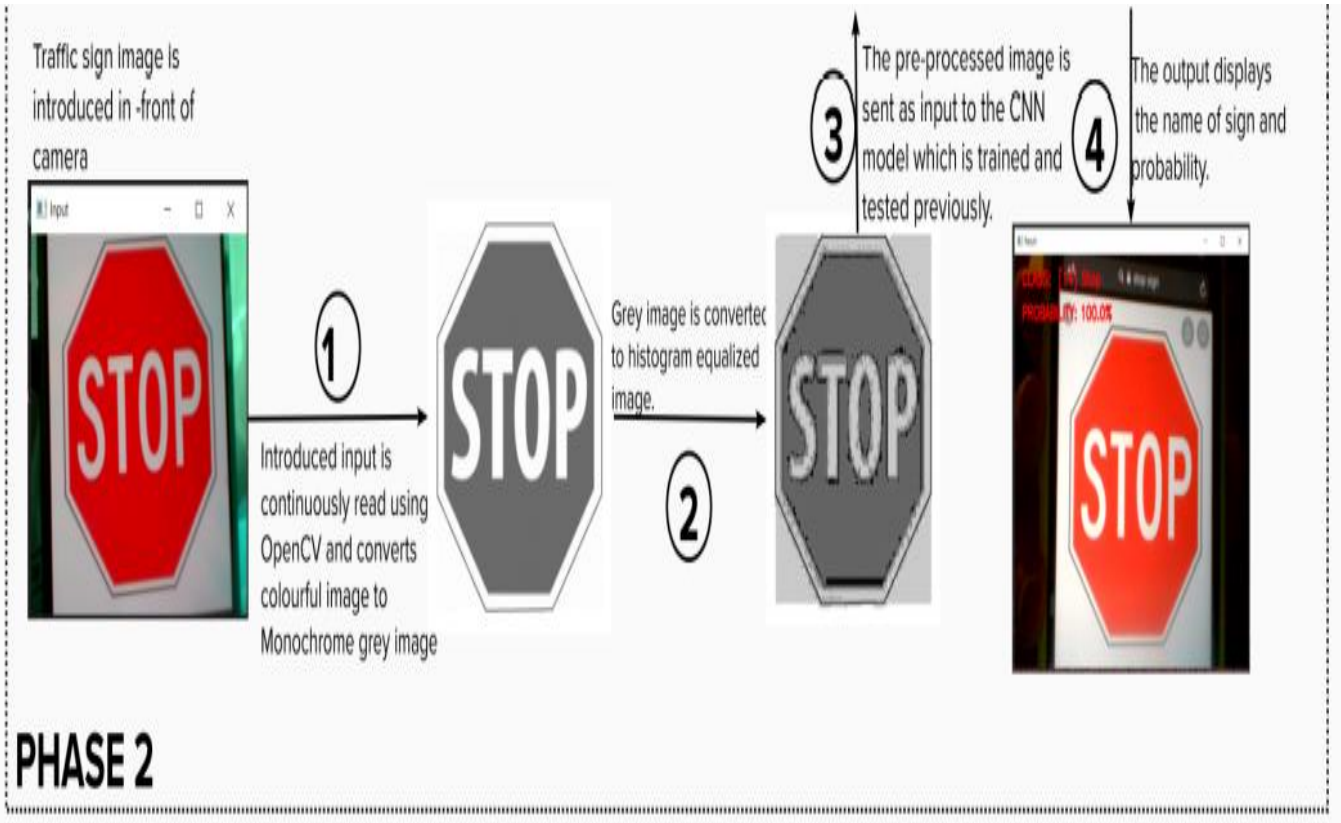
- Splitting up the images into train and test dataset.
- Pre-processing the images in respective datasets for enhancing the image quality and interpret ability.
- Building a cost-sensitive CNN model to classify the images.
- Tuning the parameters to find how significant the model's accuracy is improved.
- Utilizing this model to classify images extracted in real-time using OpenCV.
- Presenting the experiment results. The figures below summarize the steps which are conducted in the experiment to recognize traffic signs. The steps are mainly classified into two phases.


Start
Receiving Warning Message
STOP! Sign
Traffic Light
Speed Limit 30 Km/hr
Stop the car
Check if Light is Red?
NO
YES
Stop!
Check if Light is Yellow?
NO
YES
Slow Down
Check if Light is Green?
YES
Continue
Reduce the velocity of car to 30 Km/hr
Update Message Field
End

Phase 1: Training and testing the built CNN model with default parameters and then perform parameter tuning.



Phase 2: Testing the model in real-time using OpenCV.



5.1 Computer Vision:

Computer vision is the transformation of data from a still or video camera into either a decision or a new representation. It is a process that helps in understanding how images and videos are stored and how to manipulate and retrieve data from them. It is mainly used for detecting and recognizing the required data from a video (or) image. OpenCV is an open-source Computer Vision library, which is helpful in detecting and extracting the image.

5.2 Image Processing:

Image processing is a method to perform some operations on the images to enhance the image in order to scrap useful information from them. Image processing is a method in which images are sent as input, and their features or characteristics are extracted. Image processing typically has three steps:

1. Importing tools via image acquisition tools.
2. Processing the images through various processing techniques.
3. Sending an altered image or reporting that is based on image analysis as an output.

The image processing techniques used in our thesis are grayscale and histogram equalization, which are briefly explained in this thesis.

6- Environment and Libraries:

- **Python:**

  Python is an interpreted high-level, general-purpose, easily readable programming language. It is dynamically typed, and garbage collected. It supports procedural, functional and object-oriented programming. Python has a huge collection of libraries that provides tools suited to many tasks. Python is popular for Data Science, Machine Learning and Artificial Intelligence. It is also popular for web development as it contains good frameworks such as Flask, Django etc. The python libraries used in this thesis are described as follows.

- **NumPy:**

  NumPy (also called Numerical Python) is a library in python which is used for scientific computing. NumPy contains a multi-dimensional array and matrix data structures. It can be utilized to perform several mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

- **Matplotlib:**

  Matplotlib is a cross-platform data visualization and graphical plotting library in python used for plotting attractive graphs. NumPy is the required dependency for matplotlib to plot graphs. Matplotlib script is structured so that just a few lines of code are enough to generate a visual data plot in many instances. It also provides a UI menu for customizing the plots. Matplotlib is widely known and easier to visualize among other libraries that represent data.

- **Keras:**

  Keras is a Google-developed high-level deep learning API for building neural networks. It is used to make the implementation of neural networks easy. It also supports multiple back-end neural network computation. Keras is easy to learn and use, as it provides python front-end with high-level of abstraction, having multiple back-end options for computation purposes.

- **OpenCV:**

  OpenCV is an open-source library for Image processing and Computer Vision. It is used to detect and recognize entities in real-time. When it is integrated with other libraries such

as NumPy, it can preprocess the OpenCV array structure for analysis, i.e., the detected image with the help of OpenCV can be used for pre-processing using NumPy. It supports C++, Java and Python interfaces.

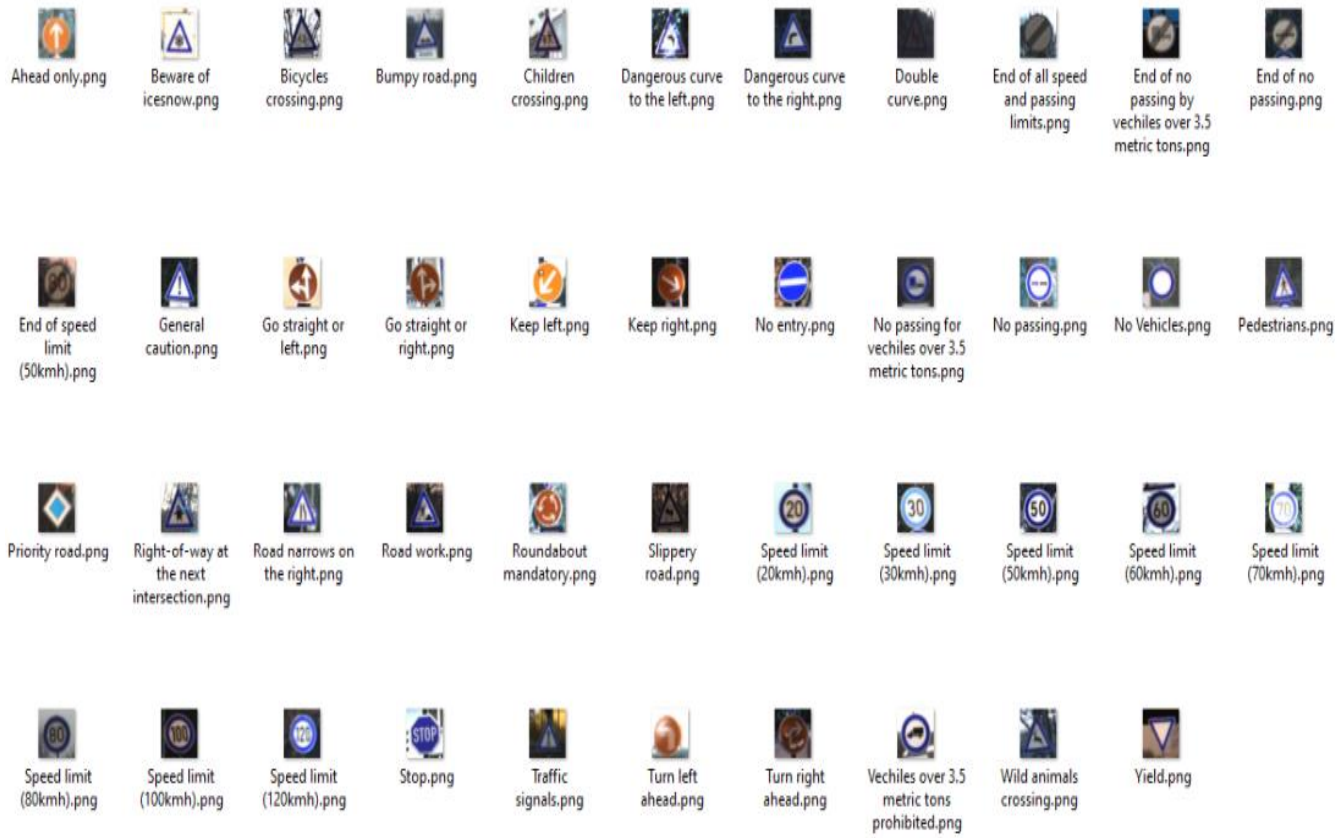
- **Pandas:**

  Pandas is an Open-source package used for Data analysis and manipulation. It is released under the BSD license. It is built on top of the NumPy package. Panda saves a lot of time by removing time-consuming and repetitive tasks associated with data.

- **Scikit – Learn:**

  Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It is built on packages like NumPy, Pandas and Matplotlib.

7- Data Overview (Traffic Signs):



8- Training and testing the model:

Training of the model is done using Adam optimizer and categorical cross entropy as loss function. Step 3 of phase 1 is done here. As shown in the Figure 4.6 the number of images is few in some classes, so data augmentation is used while training the model to generate multiple duplicate images from a single image. Fit method is used to train the model, which contains the following parameters. • Augmented images generated from flow method of image data generator class per each image in training dataset.

• **Steps-per-epoch:** Len(X_train)/ batch-size • epochs: Initially 10 epochs are taken, later can be changed for parameter tuning.

• **Validation-data:** Validation is also performed along with train, using validation dataset. After performing the training, graphs are plotted for loss and accuracy with increasing epochs among training and validation datasets. After training is completed, testing is performed with the testing dataset. These are the images that are not used for training, i.e., unseen traffic images. Later, the model is saved using the save method in'.h5' extension.

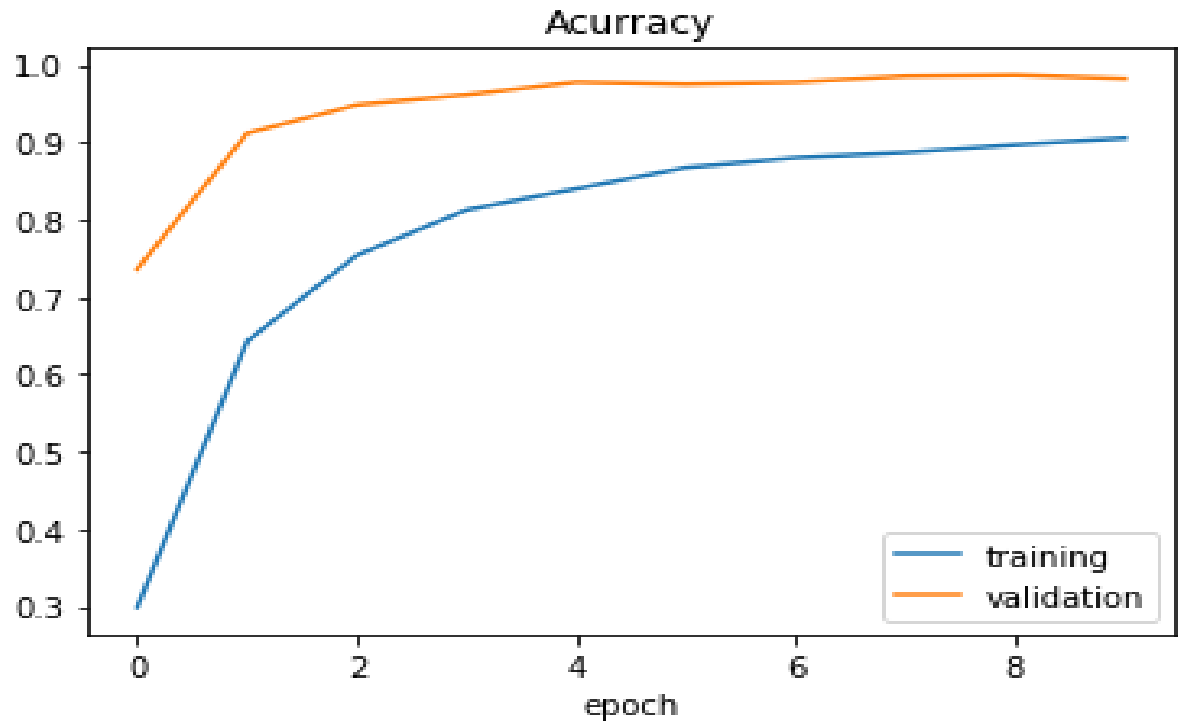
9- Distribution of dataset:

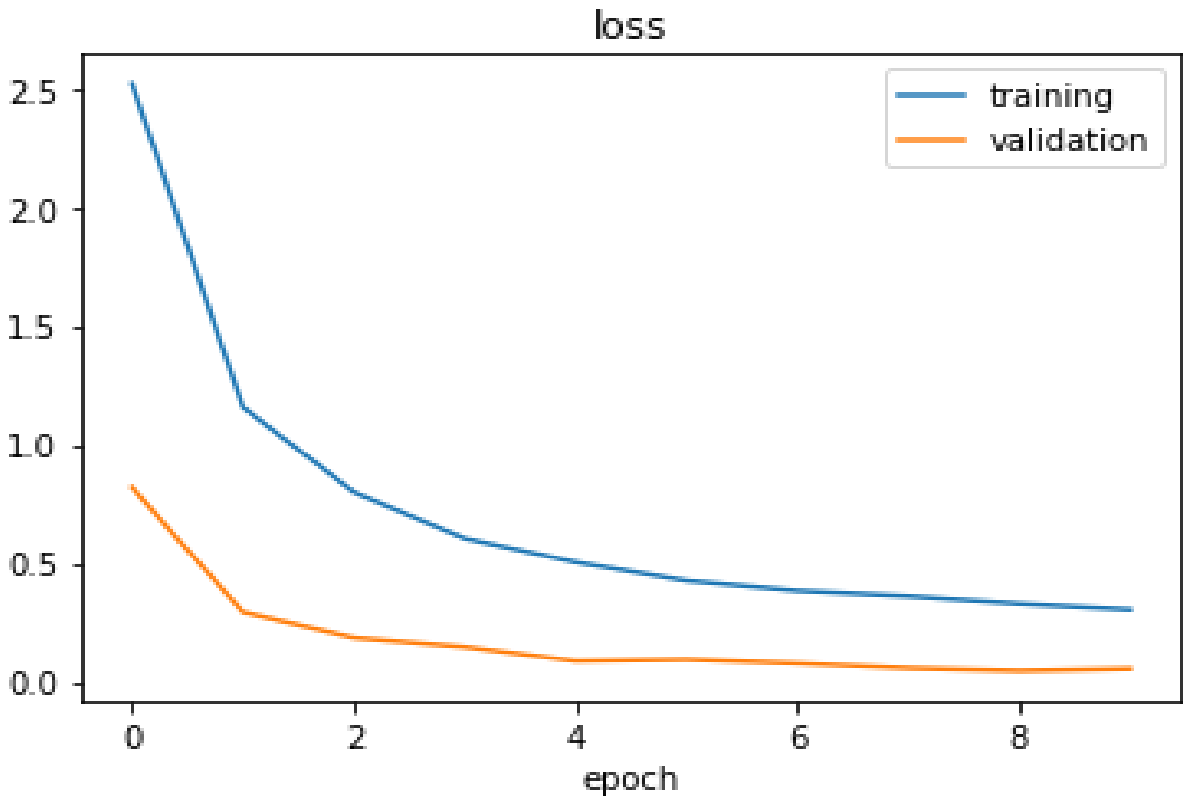
10-Results and Analysis:

By applying several learning rates and epochs:

For learning rate = 0.002 and epochs = 10

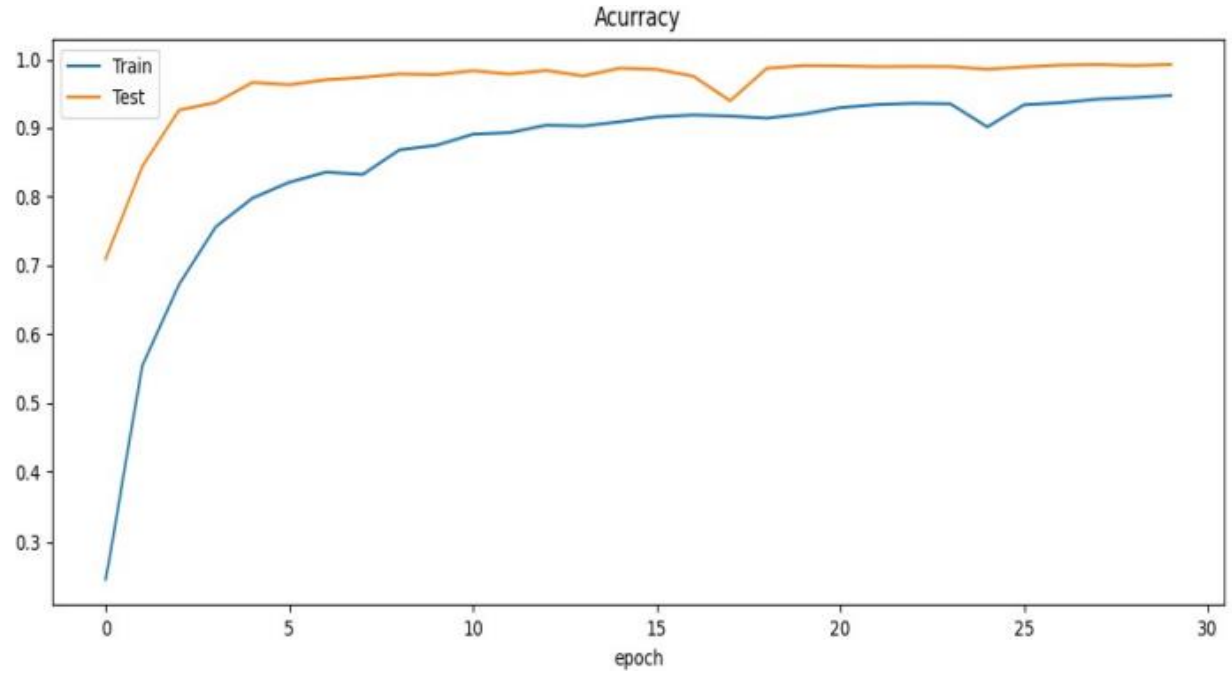
We get that Train Accuracy is = 0.8549 = 85.49%

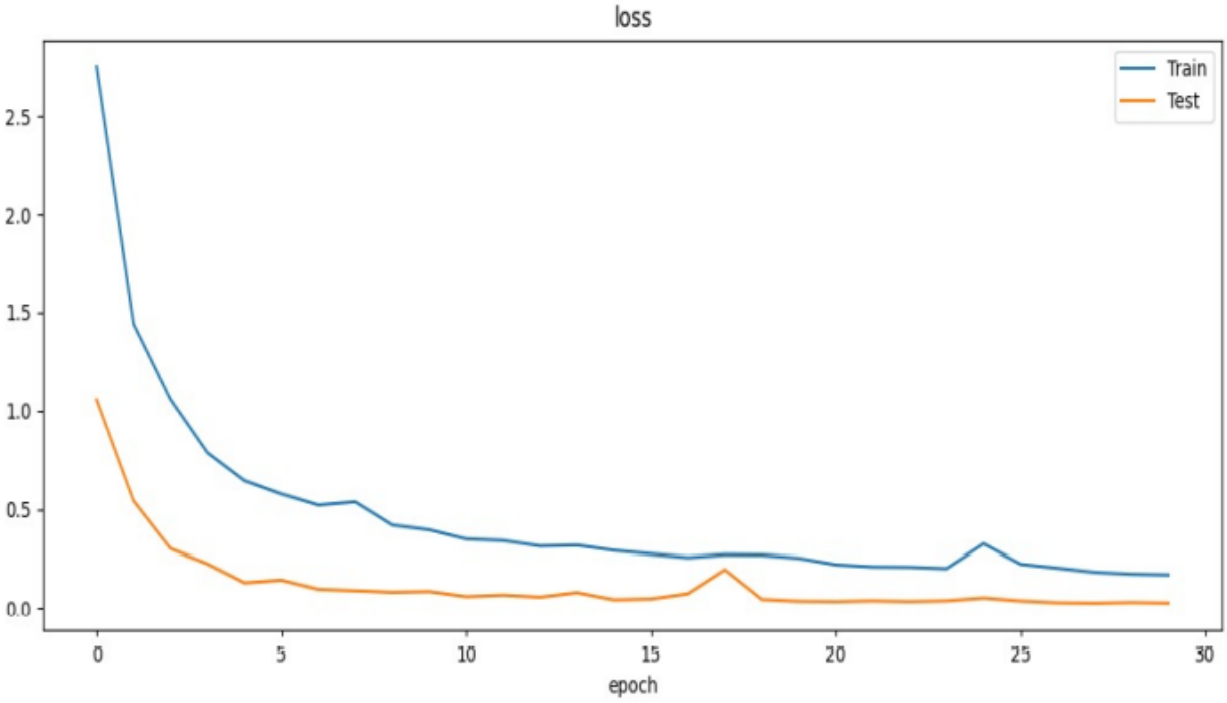




For learning rate = 0.001 and epochs = 30

We get that Train Accuracy is = 0.9510 = 95.1%





11-Using the model in Real time:

In this experiment, we tested the model on a real-time video using a webcam or any external live feed. Traffic sign images introduced to the camera are new images and are not from training or testing data. New images from each class are used for testing. We process every frame of the video to check for a traffic sign. If the probability value of the prediction is higher than the threshold value, then the class label will be displayed along with the probability. The webcam displays an example where a traffic stop sign image is placed in front of the webcam. The class name is displayed along with the probability of prediction. Each traffic sign in 43 classes is tested in real-time. While some show probability scores greater than 90%, some fluctuate their probability scores. For example, when the traffic sign 'Speed limit (20km/h)' image is placed in front of the camera, the probability score fluctuates. We tested the model with 43 different Traffic signs, one image for each class, and we noted the probabilities for each class and listed them in the table below:


SIGN: [5] Speed Limit 80 km/h
PROBABILITY: 95.09%
80


SIGN: [38] Keep right
PROBABILITY: 100.0%

Class ID	Traffic Sign	Probability
0	Speed limit (20km/h)	80-90%
1	Speed limit (30km/h)	100%
2	Speed limit (50km/h)	98%
3	Speed limit (60km/h)	99%
4	Speed limit (70km/h)	100%
5	Speed limit (80km/h)	95-100%
6	End of speed limit (80km/h)	95-98%
7	Speed limit (100km/h)	99%
8	Speed limit (120km/h)	98%
9	No passing	100%
10	No passing for vehicles over 3.5 metric tons	99%
11	Right-of-way at the next intersection	100%
12	Priority road	97-99%
13	Yield	99%
14	Stop	92-95%
15	No vehicles	91-95%
16	Vehicles over 3.5 metric tons prohibited	93-96%
17	No entry	98-99%
18	General caution	99%
19	Dangerous curve to the left	80-90%
20	Dangerous curve to the right	85-90%
21	Double curve	89-93%
22	Bumpy road	86-91%
23	Slippery road	87-90%
24	Road narrows on the right	90-93%
25	Road work	98-99%
26	Traffic signals	94-96%
27	Pedestrians	86-90%
28	Children crossing	89-94%

29	Bicycles crossing	85-90%
30	Beware of ice/snow	91-95%
31	Wild animals crossing	96-98%
32	End of all speed and passing limits	89-93%
33	Turn right ahead	95-97%
34	Turn left ahead	88-91%
35	Ahead only	99-100%
36	Go straight or right	89-92%
37	Go straight or left	85-88%
38	Keep right	100%
39	Keep left	85-88%
40	Roundabout mandatory	86-89%
41	End of no passing	90-92%
42	End of no passing by vehicles over 3.5 metric tons	87-91%

12-Conclusion:

Traffic Sign Recognition helps novice drivers through Driver Assistance System and provide safe, efficient navigation in the case of self-driving vehicles. In this thesis, we proposed a cost-sensitive CNN model in terms of computational cost, which can classify the traffic signs images from the GTSRB dataset with 95% accuracy. The proposed model architecture has 4 convolution layers which have low computational cost than various state-of-the-art architectures like VGG-16, where there are 16 convolution layers. The learning rate and number of epochs were tuned in different combinations to achieve the highest accuracy possible. Later, the model was used to predict the class labels for the traffic sign images that are introduced to the web camera with reasonably high probability. From the results, we can conclude that tuning the epoch significantly impacts the model's accuracy. The results show that changing the learning rate decreases the model's accuracy. **Hence with learning rate = 0.001 and epoch = 30, the model has achieved high accuracy, i.e., 95.1%.**

13-Future Work:

To improve the prediction probability of the images in real-time, the larger dataset can be used to train the model with more images in each class. Employing a larger dataset may result in better predictions because more features of each class label can be obtained. To solve the problem of long training times, Amazon Web Server (AWS) Deep Learning Amazon Machine Images (AMI) can be used to generate auto-scaled clusters of GPU for large scale training. One can quickly launch an Amazon EC2 instance which is pre-installed with popular deep learning frameworks and interfaces to train the model. Also, we can apply it on hardware using Raspberry Pi and using CARLA Simulator.

---

(5) GPS Feature

1- What is flutter?

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase.

First described in 2015, Flutter was released in May 2017.

Flutter uses a variety of widgets to deliver a fully functioning application. These widgets are Flutter's framework architecture. Flutter's Widget Catalog provides a full explanation and API on the framework.

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

1- Framework architecture:

The major components of Flutter include:

- Dart platform
- Flutter engine (Skia Graphics Engine)
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

2.1 Flutter has three core capabilities:

- **Fast to develop**: Build your Android and iOS applications in milliseconds with Stateful Hot Reload.
- **Expressive and flexible**: Quickly ship features with a focus on native end-user experiences.
- **Native performance on both iOS and Android**: Flutter's widgets incorporate all critical platform differences — such as scrolling, navigation, icons, and fonts — to provide full native performance.

2.2 Steps to build a flutter google maps application:

1- Set up your Flutter environment

2- Getting started

3- Adding Google Maps to the app

4-get the user’s current location

5-get the user’s drop off location

6-get direction between two points

7-draw polyline between the two points

8-enable live location

1- Set up your Flutter environment

You need two pieces of software to complete this part: the Flutter SDK, and an editor. I used Android Studio, but there are other editors to use

You can run using any of the following devices:

- A physical device (Android or iOS) connected to your computer and set to developer mode.
- The iOS simulator. (Requires installing XCode tools.)
- The Android emulator. (Requires setup in Android Studio.)

2- Getting started

3 Getting started with Flutter

The easiest way to get started with Flutter is to use the flutter command line tool to create all the required code for a simple getting started experience.

```
$ flutter create google_maps_in_flutter
Creating project google_maps_in_flutter...
[Listing of created files elided]
Wrote 127 files.

All done!
In order to run your application, type:

  $ cd google_maps_in_flutter
  $ flutter run

Your application code is in google_maps_in_flutter/lib/main.dart.
```

4 Adding Google Maps Flutter plugin as a dependency

Adding additional capability to a Flutter app is easy using Pub packages. In this step we introduce you Google Maps Flutter plugin by running the following command from the project directory.

```
$ cd google_maps_in_flutter
$ flutter pub add google_maps_flutter
Resolving dependencies...
  async 2.6.1 (2.8.2 available)
  charcode 1.2.0 (1.3.1 available)
+ flutter_plugin_android_lifecycle 2.0.3
+ google_maps_flutter 2.0.8
+ google_maps_flutter_platform_interface 2.1.1
  matcher 0.12.10 (0.12.11 available)
  meta 1.3.0 (1.7.0 available)
+ plugin_platform_interface 2.0.1
+ stream_transform 2.0.0
  test_api 0.3.0 (0.4.3 available)
Downloading google_maps_flutter 2.0.8...
Downloading flutter_plugin_android_lifecycle 2.0.3...
Changed 5 dependencies!
```

3- Adding Google Maps to the app:

It's all about the API keys

To use Google Maps in your Flutter app, you need to configure an API project with the Google Maps Platform, following the Maps SDK for Android's Using API key, Maps SDK for iOS' Using API key, and Maps JavaScript API's Using API key **[11]**. With API keys in hand, carry out the following steps to configure Android application:

Adding an API key for an Android app

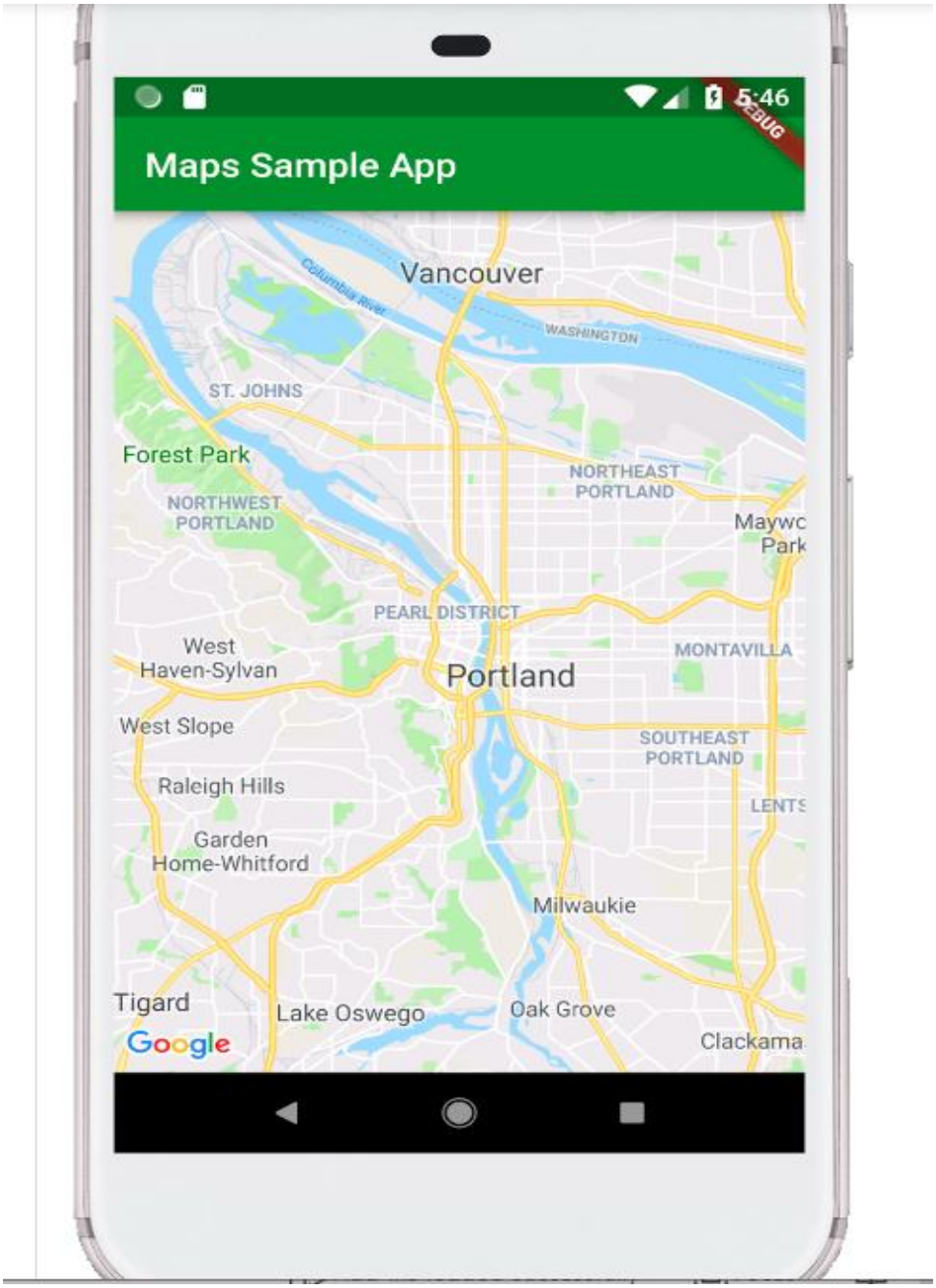
To add an API key to the Android app, edit the `AndroidManifest.xml` file in `android/app/src/main`. Add a single `meta-data` entry containing the API key created in the previous step inside the `application` node.

Putting a map on the screen

Now it's time to get a map on the screen. Update `lib/main.dart`

Running the app

And here is a sample of the implementation of the google maps in the application



4- Get the user's current location:

To get current location in flutter we will make use of 2 flutter package.

1. Geolocator: To get user's device present location Latitude & Longitude in flutter.
2. Geocoding: To get location address using late & long coordinates (placemarkfromcoordinates) (late long to address)

Flutter geolocator

Features:

- Get the last known location.
- Get the current location of the device.
- Get continuous location updates.
- Check if location services are enabled on the device.
- Calculate the distance (in meters) between two geocoordinates.
- Calculate the bearing between two geocoordinates

Flutter geocoding

Geocoding refers to transforming street address or any address into latitude and longitude. Reverse Geocoding refers to transforming latitude and longitude into its corresponding street address

Address class helps in fetching the street address, locality, sub-locality, city, country, landmark etc. features of the location.

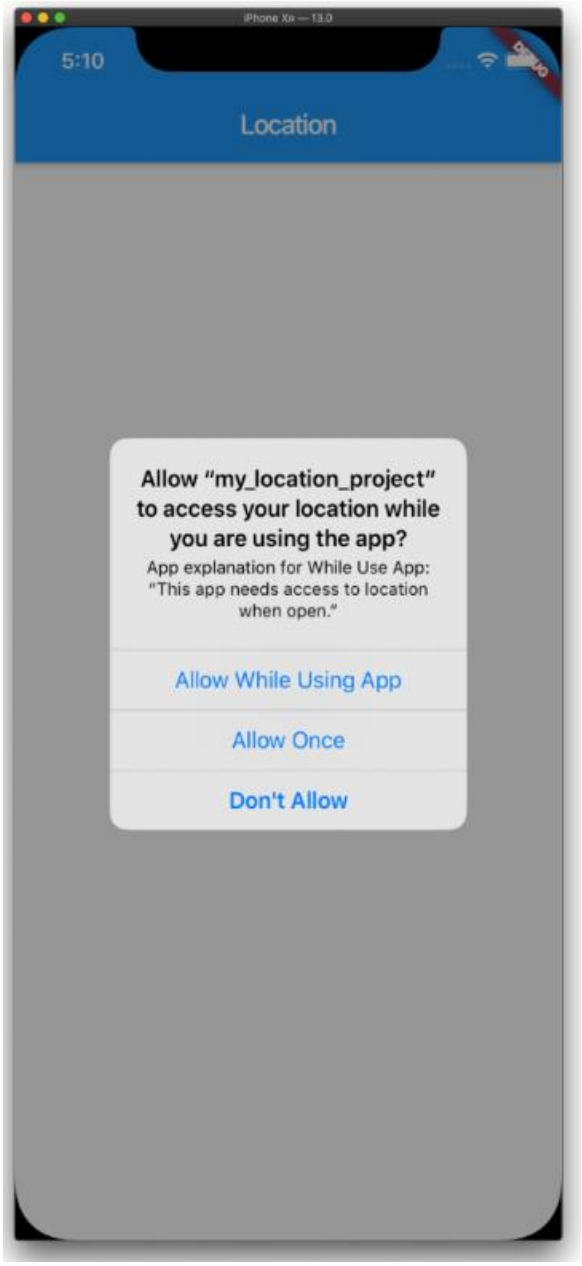
To get the user's current location we need to follow these steps:

1-Getting the Latitude and Longitude

You can accomplish that by creating an instance of Geolocator and calling getCurrentPosition. This should ask the user whether they are interested in using the Location feature and if so, get the current location as a Position.

And here is an example of the screen

(a)

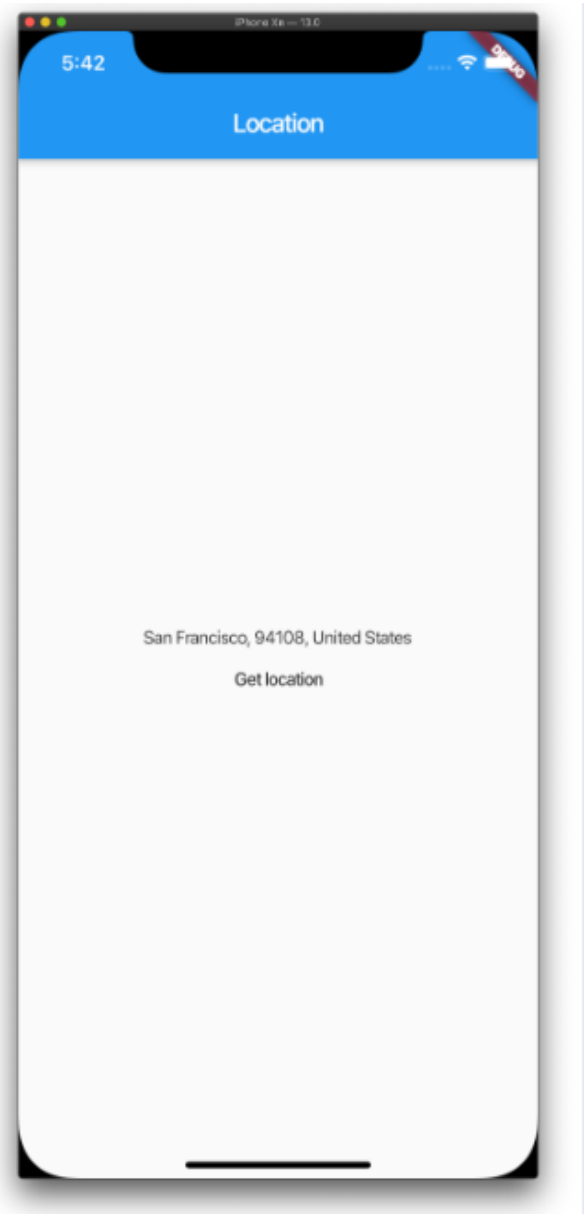


(b)

2- Converting Latitude and Longitude to a Human-readable Address

The next step is converting the coordinates to display an address.

Passing latitude and longitude coordinates to placemarkFromCoordinates will return a Placemark. Placemark contains information like locality, postalCode, and country.



5-get the user's drop off location

1- First, we are going to use Place Autocomplete API from google console map

Place Autocomplete:

The autocomplete service in the Places SDK for Android returns place predictions in response to user search queries. As the user types, the autocomplete service returns suggestions for places such as businesses, addresses, plus codes, and points of interest.

We added an autocomplete widget

The autocomplete widget is a search dialog with built-in autocomplete functionality. As a user enters search terms, the widget presents a list of predicted places to choose from. When the user makes a selection, a `Place` instance is returned, which your app can then use to get details about the selected place.

1- Secondly, we are going to make the list view clickable and whenever the use select any place from this list, we have to get the coordinates of that place (latitude, longitude)

Place Details API

Once you have a place_id from a Place Search, you can request more details about a particular establishment or point of interest by initiating a Place Details request. A Place Details request returns more comprehensive information about the indicated place such as its complete address, phone number, user rating and reviews.

6-Get direction between two points:

In this step we are going to draw a polyline between the pickup location (current location) to the drop off location using google direction API

Why use Directions?

With Directions, you can retrieve more than simple driving directions. You can get directions for several modes of transportation, such as transit, driving, walking, or cycling.

To use Directions functionality and get serialization of requests and deserialization of responses, automatic retries, and some client validation of requests, try one of our client libraries.

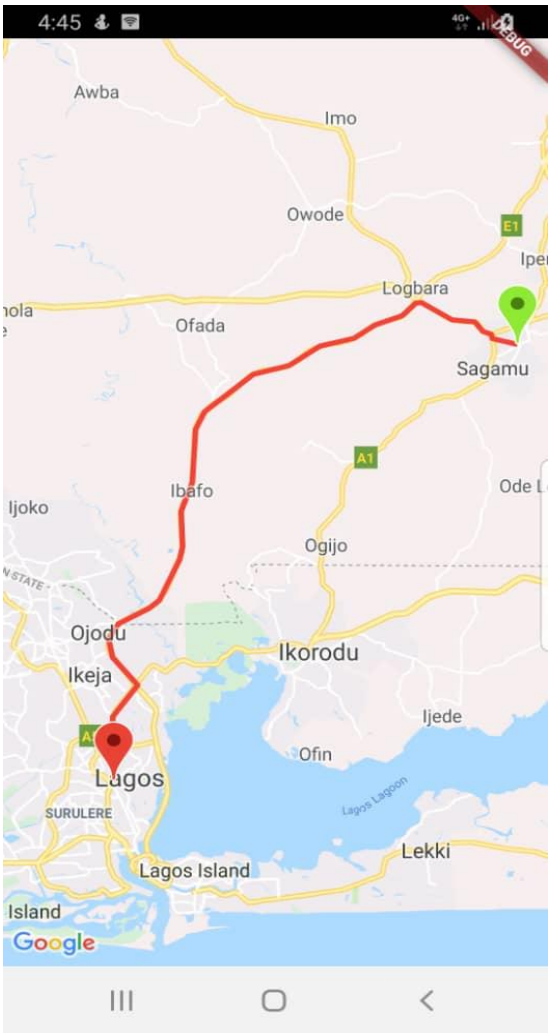
If you want direction calculations that respond in real time to user input (for example, within a user interface element), then use the Directions API. If you're already using the Maps JavaScript API, then use the Directions service to get the Directions API's functionality.

7-Draw polyline between the two points

In this step we have to get the encoded points first and then translate these encoded points on map as a polyline point that is from the pickup location to the drop off location, so we have to implement the flutter polyline points package

A flutter plugin that decodes encoded google polyline string into list of geo-coordinates suitable for showing route/polyline on maps

This package contains functions to decode google encoded polyline string which returns a list of co-ordinates indicating route between two geographical positions



(6) Communications Protocols

Introduction:

Communication Protocols are a set of rules that allow two or more communication systems to communicate data via any physical medium. The rules, regulations, synchronization between communication systems, syntax to be followed and semantics are all defined by the term protocol. Protocols can be implemented by both hardware and software or combination of both. Analog and digital communication systems use various communication protocols widely. In addition, each protocol has its own application area.

Each communicating entity should agree to some protocol to exchange information. Many different protocols are available for embedded systems and are deployed depending upon the application area.

1- Communication Protocol Types:

There are many types of communication protocols each one has a different configuration and applications.

1-1 SPI (Serial Peripheral Interface) Protocol:

The SPI is a full duplex protocol which uses master slave configuration to establish communication. Motorola invented this protocol in 1980. SPI is used in microcontrollers with interfaces like EEPROM, LCD displays, etc.

The master device has to select the slave to which the data needs to be sent using the Slave Select pin by activating a low input signal. Then the clock pulse from SCLK pin will be sent to slave to synchronize the transmission. The data from Master pin goes through MOSI pin and from slave to master comes through MISO pin.

1-2 I2C (Inter-Integrated Circuit) Protocol:

I2C is a two-wire serial bus communication protocol invented in 1982 by Philips Semiconductors. It is a two-wire communication interface and commonly used to connect low-speed devices like microcontrollers, A/D and D/A converters, I/O devices and other peripherals in embedded systems.

In I2C protocol each device is given a specific address. When a master wishes to transfer data to a slave, it sends the address of slave device through SDA pin for which the message is intended to. It is followed by Data frame and only the slave with the specified address will accept this incoming data.

1-3 USB (Universal Serial Bus) Protocol:

USB is one of the well-developed and most used communication protocol. Despite having different standards and hardware types, it is considered to be versatile and effective in embedded systems. USB communication takes place by means of polling where the host device initiates the data transfer. When a device connects to the host it assigns address to the system and uses it to perform data transfer.

Every data transfer in a USB communication is done by means of data packets, generally there are three components in a data packet: a token packet which carries information such as data type, device address. Next the data packet which holds the actual data to be transferred and EOP to indicate the data packet is ended.

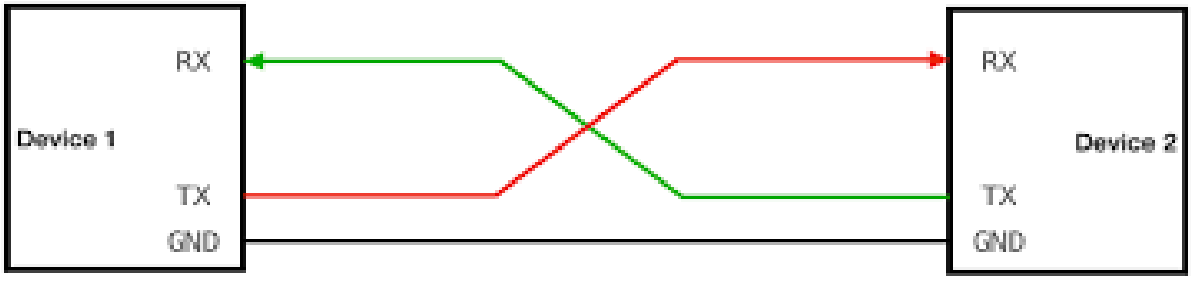
1-4 UART Protocol:

UART (Universal Asynchronous Receiver/Transmitter) Protocol is a serial communication protocol developed by Digital Equipment Corporation in 1960's. It supports Full duplex, half duplex and simplex communication. This is one of the most popular protocols and will be in almost all microcontrollers.

The communication in UART takes place through two pins.

- **RX:** Receive the incoming data
- **TX:** Transmits the outgoing data

UART is two-wired communication, the serial data is handled by Tx (Transmitter) and Rx (Receiver) pins.

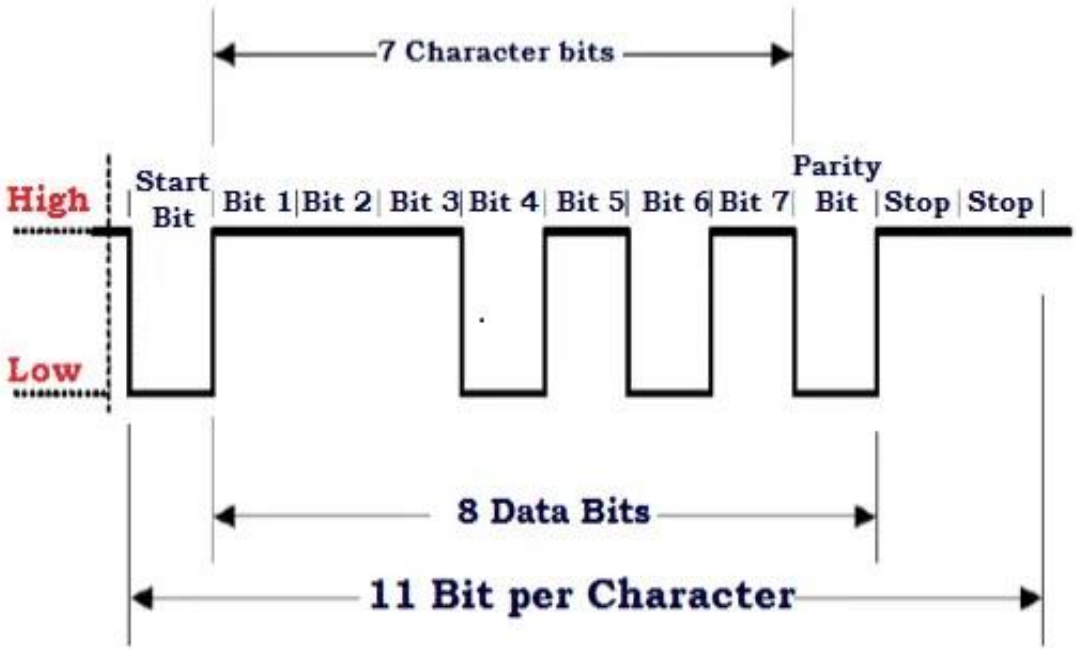


UART transmits data asynchronously, which induces that no clock signal is associated in transmitting and receiving data. Instead of clock signal, UART embed start and stop bits with actual data bits, which defines the start and end of data packet.

When receiver end detects the start bit, it starts to read the data bits at specific baud rate meaning both transmitting and receiving peripherals should work under same baud rate. UART works under half duplex communication mode meaning it either transmits or receives at a time.

Data transmission in UART takes place by means of data packets which usually consists of 8 data bits, start bits and parity bits to perform error correction.

The data frame starts with a start bit, followed by data bits and then by parity and stop bit. Most microcontrollers provide ways to configure the number of data bits and composition of the data frame.



The advantages of UART Communication Protocol:

- Clock signal is not required
- Cost effective
- Uses parity bit for error detection
- Requires only 2 wires for data communication

2- UART Configurations:

Asynchronous data transfer has a protocol, which is usually as follows:

The first bit is always the START bit (which signifies the start of communication on the serial line), followed by DATA bits (usually 8-bits), followed by a STOP bit (which signals the end of data packet). There may be a Parity bit just before the STOP bit. The Parity bit was earlier used for error checking.

- **Start bit:** To declare the start of transmission, always low '0'.
- **Data bits:** 5, 6, 7, 8 or 9 bits of useful data bits.
- **Parity bit:** To check for transmission errors.
- **Stop bit:** one or two stop bits to declare end of frame, always high '1'.

2-1 UBRR Register:

This register is used to set the baud rate value by using this equation:

$$\boldsymbol{Baud\ Rate} = \frac{\boldsymbol{fosc}}{\mathbf{16}(\boldsymbol{UBRR} + \mathbf{1})}$$

So, to get certain value of baud rate we need to write in UBRR value make the result as we wanted.

19.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRnH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits, and the UBRRnL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

2-2 UCSRA Register:

UART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bit 7: UART Receive Complete

Bit 6: UART Transmit Complete

Bit 5: UART Data Register Empty

Bit 4: Framing Error

Bit 3: Overrun

Bit 2: Reserved Bit

Bit 1: Double the UART Transmission Speed

Bit 0: Multi-processor Communication Mode

2-3 UCSRB Register:

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7: UART Receive Complete Interrupt Enable

Bit 6: UART Transmit Complete Interrupt Enable

Bit 5: UART Data Register Empty Interrupt Enable

Bit 4: Receiver Enable

Bit 3: Transmitter Enable

Bit 2: 9 Bit Characters

Bit 1: Receive Data Bit 8

Bit 0: Transmit Data Bit 8

1-2-4 UCSRC Register:

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Bit 7: URSEL – USART Register Select

Bit 6: UMSEL – USART Mode Select

UMSEL	MODE
0	Asynchronous
1	Synchronous

Bit 5, 4: Parity Mode

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reversed
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Bit 3: Stop Bit Select

USBS	Stop Bit(s)
0	1-bit
1	2-bits

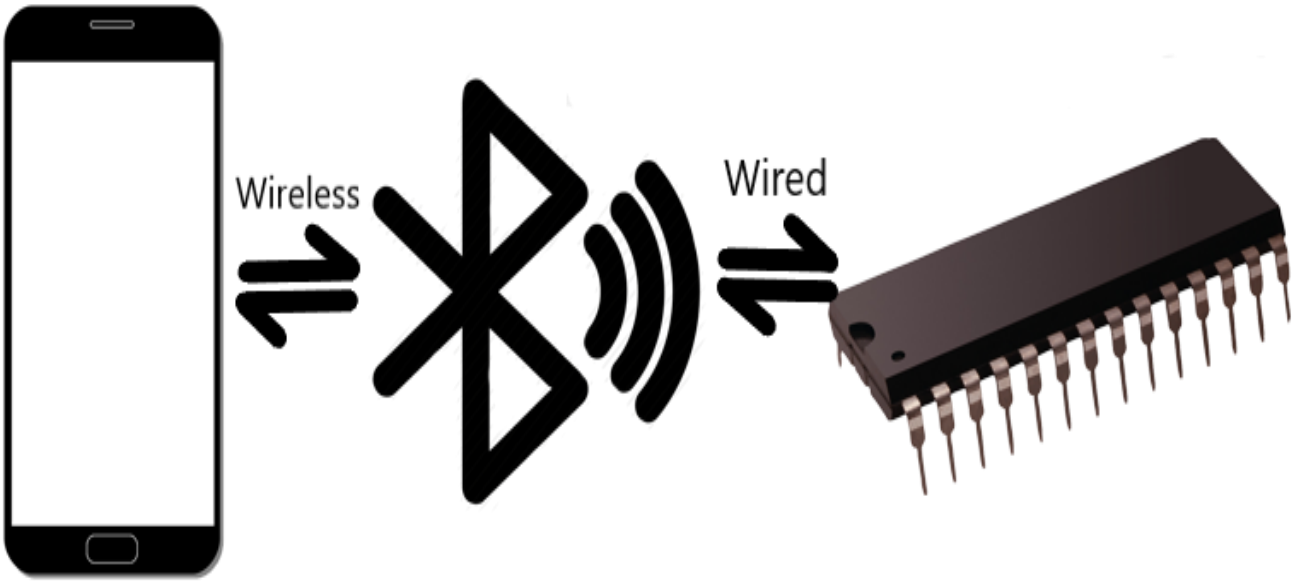
Bit 2, 1: Character Size

UCSZ2	UCSZ1	UCSZ0	No. of bits
0	**0**	**0**	**5-bits**
0	**0**	**1**	**6-bits**
0	**1**	**0**	**7-bits**
0	**1**	**1**	**8-bits**
1	**0**	**0**	**Reserved**
1	**0**	**1**	**Reserved**
1	**1**	**0**	**Reserved**
1	**1**	**1**	**9-bits**

Bit 0: Clock Polarity

3- Communication with Mobile App:

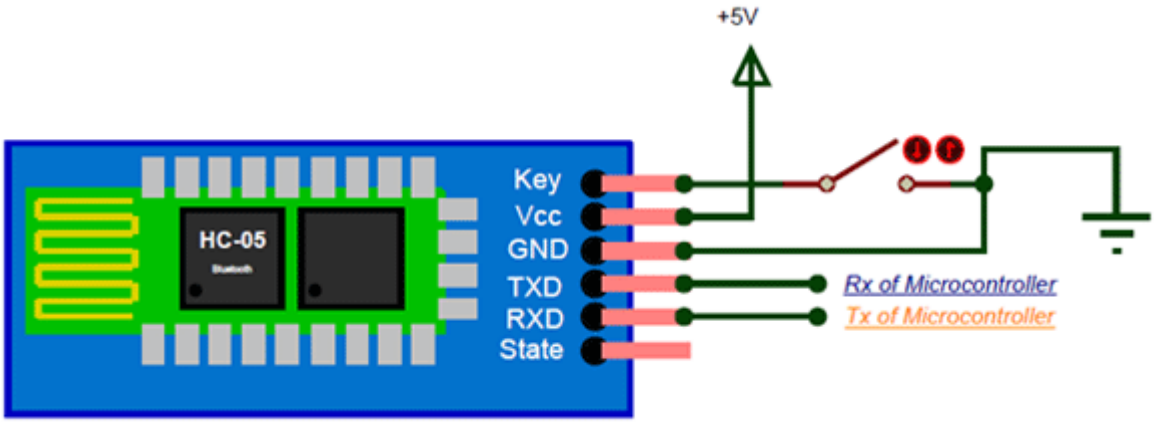
The Microcontroller couldn't communicate wireless with the App so there is a need to use a **Bluetooth Module** which will communicate with the app through Bluetooth and then communicate with microcontroller wired by UART.



3-1 Bluetooth Module HC-05:

It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications. It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions. It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air. It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

Pin Description



Key/En: It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is **38400bps and 9600** in data mode.

HC-05 module has two modes,

1. **Data mode:** Exchange of data between devices.

2. **Command mode:** It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

VCC: Connect 5 V or 3.3 V to this Pin.

GND: Ground Pin of module.

TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

State: It tells whether module is connected or not.

Pair HC-05 and smartphone:

1. By search for new Bluetooth device from the phone will find Bluetooth device with "HC-05" name.
2. Clicking on connect/pair device option; default pin for HC-05 is 1234 or 0000.

After pairing two Bluetooth devices, open Bluetooth terminal application and connect to paired device HC-05.

It is simple to communicate, we must type in the Bluetooth terminal application of smartphone. Characters will get sent wirelessly to Bluetooth module HC-05. HC-05 will automatically transmit it serially to the PC, which will appear on terminal. Same way we can send data from PC to smartphone.

Each character considers as an identification so used to activate certain task by microcontroller.

Participation:

1. **(Self-Parking) Feature:** Applied as Software and Hardware.

   (Ashraf Mohamed)

   - DIO Driver
   - Timer Driver
   - Motors Driver
   - Ultrasonic Driver

2. **(Smart Traffic Sign Recognition) Feature:** Applied as Software using Python and some AI libraries. [Image Processing]. **(Ahmed Hesham)**
3. **(GPS) Feature:** Applied as Software. **(Sara Yasser)**
4. **(Adaptive Cruise Control) Feature:** Applied as Software and Hardware. **(Muhammed Gamal El-Din, Fady Ibrahim)**
5. **OVERTAKE Feature:** Applied as Software and Hardware.

   (Muhammed Gamal El-Din)

6. **Communication Protocols:** UARTT Driver **(Fady Ibrahim)**

References:

[1]https://roundtable.menloschool.org/issue20/6_Chen_MS_Roundtable20_Winter_2015.pdf

[2] https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

[3] https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

[4] https://www.teachmemicro.com/lm393-ir-module-motor-speed-sensor/

[5] H. Soma, Y. Shiraishi, T. Watanabe, Y. Takada, and Y. Takae, "Trust in low-speed adaptive cruise control systems—analysis of trust structure," Review of Automotive Engineering, vol. 26, no. 2, pp. 211–212, 2005

[6] https://ieeexplore.ieee.org/abstract/document/9034121

[7] https://www.sciencedirect.com/science/article/pii/S2666691X20300300

[8] https://www.quora.com/Does-increasing-epochs-increase-accuracy

[9] P. Dhar, M. Z. Abedin, T. Biswas, and A. Datta, "Traffic sign detection — a new approach and recognition using convolution neural network," in 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2017, pp. 416–419.

[10] A. Lorsakul, and J. Suthakorn, "Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV", The 4th International Conference on Ubiquitous Robots and Ambient Intelligence, 2007.

[11] Y. Ye, GPS Controlled Autonomous Vehicle: An Interesting Approach to GPS Guided Autonomous Vehicle Navigation LAP LAMBERT Academic Publishing, 2011