Abstract

Final Report – Automated Smart Car

Talha Ahmed Khan
Jafar Abbas
Simaar Ahmad
Muneeb Baig

For the past hundred years, cars have become cleaner, safer, more affordable and efficient. Despite these improvements, motor vehicle deaths have been very high: "On average in 2012, 92 people were killed on the roadways of the U.S. each day, in 30 800 fatal crashes during the year"[1].

Inspired by Google and Tesla, car manufacturers are all working on their own versions of an autonomous vehicle to help get these numbers down and for other benefits such as saving millions of hours in traffic jams, reducing wasted urban space reserved for parking lots, convenience for travel and efficient goods transportation.

The challenge of building a driverless car is immense because of the safety-critical environment in which vehicles operate and the technology involved. The goal of our project is to bring to life a small- scale smart car equipped with auto-pilot: allowing safe travel from one point to another without outside intervention.

.

Final Report

Automated Smart Car

Team #18

A Report Presented to
The Department of Electrical & Computer Engineering Concordia University

In Partial Fulfillment of the Requirements of ELEC/COEN 490

By

Jafar Abbas ID: 26346650
Talha Ahmed Khan ID: 26734197
Simaar Ahmad ID: 27192258
Muneeb Baig ID: 26836054

Project Supervisor
Dr. Kash Khorasani

Table of Contents

List of Figures

List of Tables

1. Introduction

This project focuses on design, implementation and testing of the autonomous car on a small scale. The car would be able to use Autopilot functionality with ultrasonic sensors, gyroscope, speed encoder and image processing as our sensing technologies to move from one point to another with minimal manual intervention. Self-driving car gives passengers the ability to travel safe without getting into accidents. Smart car would be beneficial for visually impaired and disabled people by providing them the facility to travel without any ones assistance.

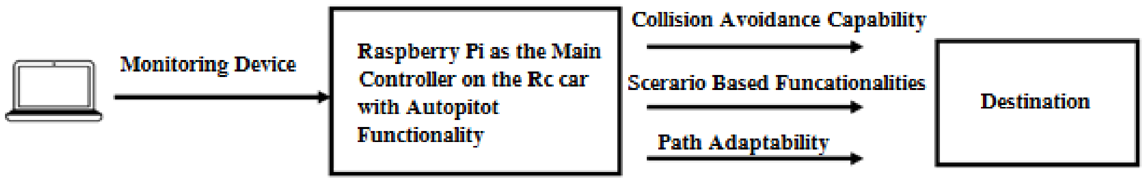

Monitoring Device
Raspberry Pi as the Main Controller on the Rc car with Autopitot Functionality
Collision Avoidance Capability
Scerario Based Funcationalities
Path Adaptability
Destination

Figure 1.1: Black box representation of the system

2. Project Overview

2.1 Functional Requirements

- Align the car to keep it between two lanes
- Collision Avoidance System to guide the car when it encounters an obstacle
- Perform main functionalities of driving like maintaining speed, accelerating, decelerating, changing lanes, turning
- Ability to stop at intersections
- Drive from point A to point B with minimal manual intervention

2.2 Non Functional Requirements

- Car should obey safe driving standards
- The car should have manual override
- The car must be able to drive in an indoor track
- Budget of this project must be within $400

2.3 System Block Diagram

Based on our requirements, we have identified blocks that will be required in our system and loosely mapped their interactions in the following diagram;

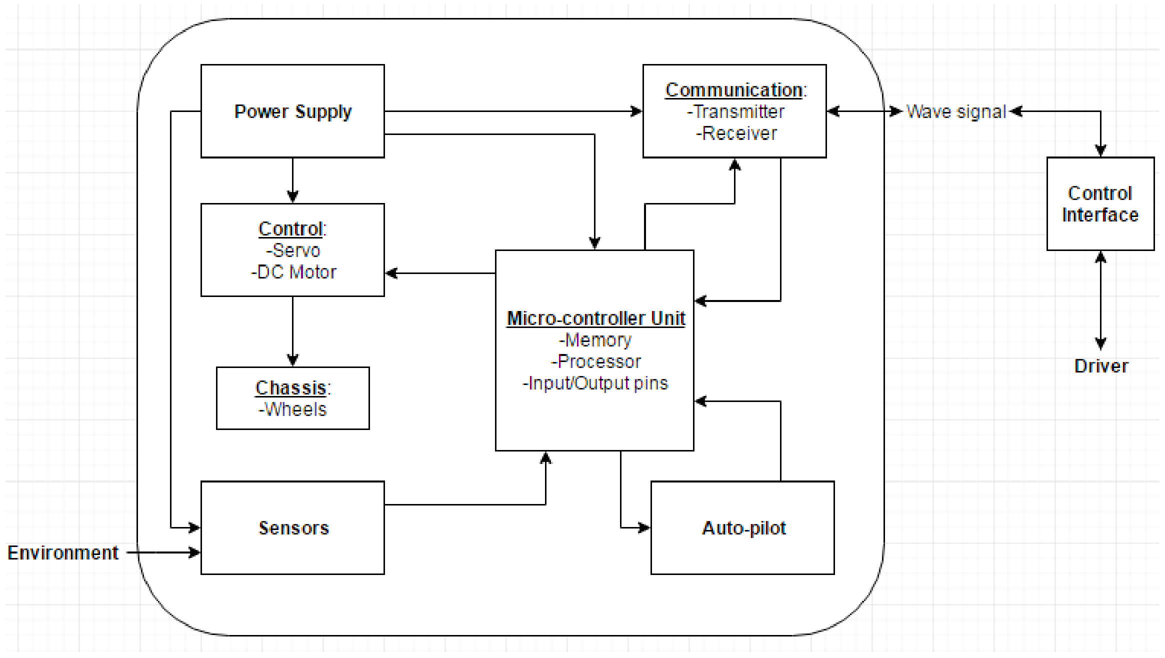


Figure 2.1 System Block Diagram

2.4. System Specifications

The following table of specifications are preliminary, based on rough calculations, estimates and similar solutions researched. They are system-wide guidelines are will be refined over time, as design decisions are made.

2.4.1. Design Specifications:

Parameter	Min	Typical	Max	Units
Width of proving ground	-	245	-	cm
Length of proving ground	-	304	-	cm
Width of Lane				
Length of car	-	36	-	cm
Width of car	-	20	-	cm
Operating speed of the car	12.08	13.5	14	cm/s
Braking distance from obstacle	15	20	25	cm
Gap between car and lanes	4	5	5.5	cm
Payload	0.35	0.4	0.5	kg
Battery life	10	15	20	min
Battery charging time	110	120	125	min
Range of controllers	5	10	15	m

Table 2.1: System Specifications

3. Final Design

The final design comprises of both system level and component level design approaches that were used in Phase 1 and Phase 2 respectively.

3.1 System Level

In Phase 1 we had proposed a design of our project on a system level. We had evaluated different methods and approaches in which our project can be done and had concluded back then that the best approach was,

- Method of processing – Onboard Processing
- Method of sensing – Combination of Sensors and Image processing
- Method of tracking – Internal map
- Method of controller – State-machine algorithm
- Car size – 1: (1/10)

3.2 Component Level

The following table shows the list of components used in the autonomous RC car, highlighting important specifications.

Component	Description	Specifications	Dimensions (LxWxH) / mm	Weight / g
Raspberry Pi 3B	Used to process all data and control all components of the RC car; acts like a brain of the system	• Required Power: 5V, 2.5A • 28 GPIO programmable pins • Voltage Output: 3.3V • Current Output: 16mA (each GPIO) • CPU: 1.2GHz • Memory: 1GB	85 x 49 x 4	42
SINOELE Mini Power Bank	Used to supply power to the Raspberry Pi	• Voltage Output: 5V • Current Output: 2.5A	78 x 78 x 23	181
WP-1040-BRUSHED ESC	Used to control the speed of a motor	• Battery: 2-3S LIPO or 7.4V NiMH • FWD Cont. Current: 40A • BWD Cont. Current: 20A	46.5 x 34 x 28.5	65
540 Brushed Motor	Used for moving forward and reverse	• Operating Voltage 7.2V • Rated load speed: 13600 RPM	Length: 56.45 Diameter: 36 Shaft: 3	158

Component	Description	Specifications	Dimensions (LxWxH) / mm	Weight / g
HexFly HX-3CP Servo	Used for turning left and right	• Operating Voltage: 4.8V • Max. Voltage: 6V • Torque at 4.8V: 3.8 kg cm • Torque at 6V: 4.3 kg cm	40 x 20 x 36	38
HC-SR04 Ultrasonic sensor	Used to detect obstacles from a distance	• Operating Voltage: 5V • Working Range: 2cm – 400cm • Measuring Angle: 15°	40 x 20 x 15	8.5
Hall sensor	Used to time the speed of the wheels (calculate distance travelled)	• Operating Voltage: 5V • Working Range:	20 x 20 x 2	9
SainSmart Camera Module	Used for detecting lanes (image processing)	• Camera: 5MP • Video: 1080p30, 720p60 and 640x480p60/90	24 x 25 x 2	3

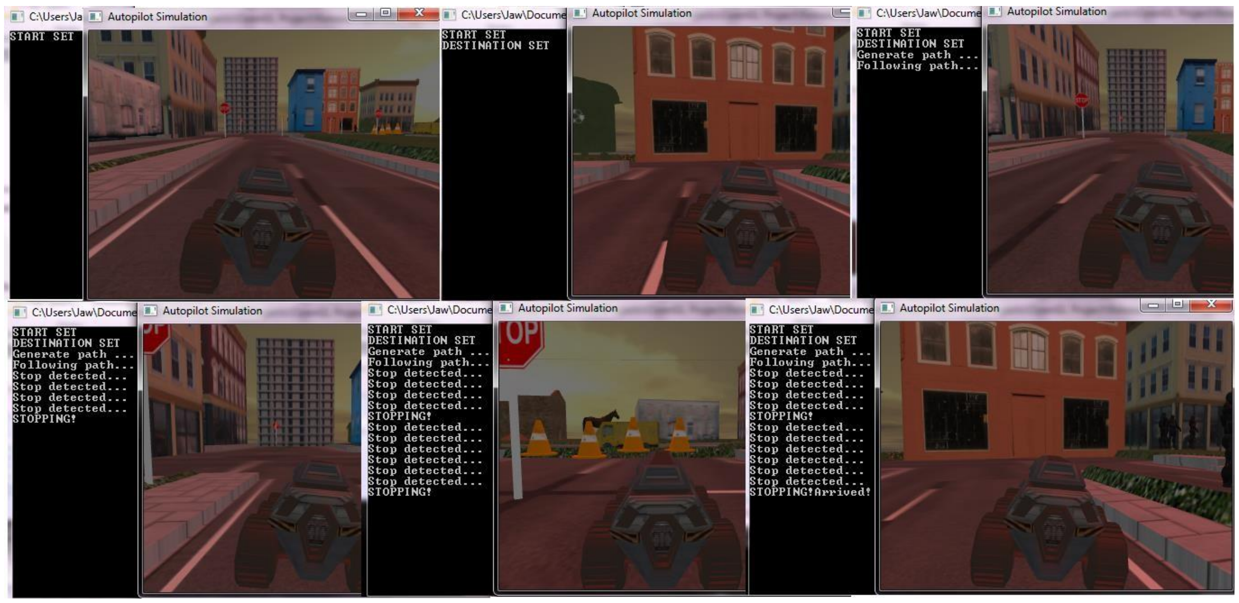
Table 3.1 Component Specifications

4. Simulation Results

Before implementing image processing and autopilot on the RC car we simulated our design plan on Open GL and Open CV. Following are the main sections that we simulated:

4.1 Auto-Pilot

Briefly, since our map consists of coordinates, we use the position of the camera to set the starting position and destination with 2 different keyboard buttons. Then another button computes the path. The path consists of a sequence of coordinates; hence we can simulate driving along this path by moving the car point by point.
We can visually see the car travelling along the computed path. We can also directly test the stop sign implementation.



```
START SET
```

```
START SET
DESTINATION SET
```

```
START SET
DESTINATION SET
Generate path ...
Following path...
```

```
START SET
DESTINATION SET
Generate path ...
Following path...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
STOPPING!
```

```
START SET
DESTINATION SET
Generate path ...
Following path...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
STOPPING!
Stop detected...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
STOPPING!
```

```
START SET
DESTINATION SET
Generate path ...
Following path...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
STOPPING!
Stop detected...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
Stop detected...
STOPPING!Arrived!
```

Figure 4.1: Testing of path finding and stop sign implementation

We output text in the console to show what is happening. In the above picture, the first frame reads “START SET”, then “DESTINATION SET”, then “Generating path…”, “Following path…”, then when it approaches both stop signs in the way it keeps repeating “Stop detected…” until it reaches it and says “STOPPING!”. When the destination is reached, it simply prints “Arrived!”.

The simulation is successful. Source and destinations are able to be set and a correct path is generated 99% of the time. Occasionally the path finding algorithm malfunctions for an unknown reason. The stop signs logic has a bug that occasionally creeps up, seemingly caused by a poorly placed red line in the stop-sign image.

4.2 Image Processing

The system consists of the four subsystems which includes video camera, image processing, controller and vehicle movement as an output resulted due to the image processing and controller.

The most important aspect of this system development is image processing which is to be used for locating the position of the vehicle with respect to the boundaries of the road. In Phase 2 of the project we worked on the detection of the desired lanes so that we can know the borders of the roads. The process was based on the real time data of video sequences obtained from the webcam of the laptop.

Image Processing completed up till phase 2 was a three stage process that incorporates the following processes:

1) Canny Edge Detection
2) Hough Transformation
3) Lane Tracking

1) Canny Edge Detection:
   It was used to extract the edges of the images and to remove the image noise. It helps us discard the edges which we don't need to use for our processing.

2) Hough Transformation:
   Canny edge detection done in the first process was used as an input to the Hough transformation. Hough lines was used in Hough Transformation to draw the lines on the edges. The lane boundaries was made possible with the help of Hough Transformation.

3) Lane Tracking:
   This was done possible by analyzing each image fetched from the video camera and then implementing Canny Detection and Hough Transformation. It determines the location of lane boundaries in a sequence of consecutive images using the data of the previous images in the sequence.

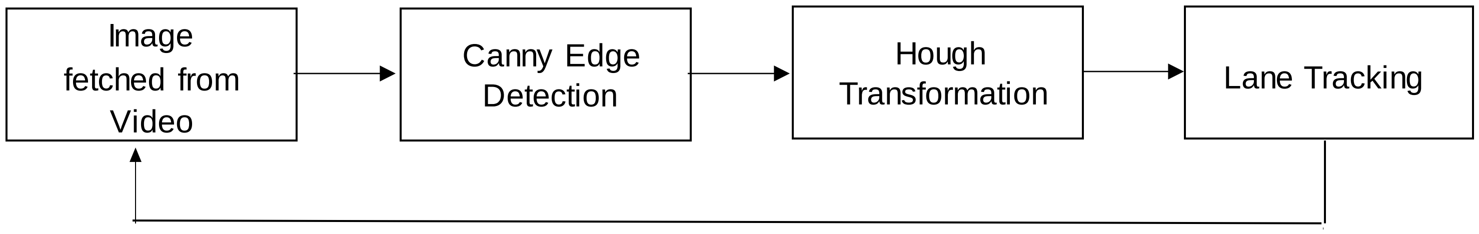


Figure 4.2 Block diagram of Image Processing procedure:

The simulation was performed using Microsoft Visual Studio and OpenCV dependent libraries. Initial simulation was done on a static picture to test the functionality of Canny Edge Detection and Hough Transformation.
Following were the results that we obtained after performing image processing on still image.

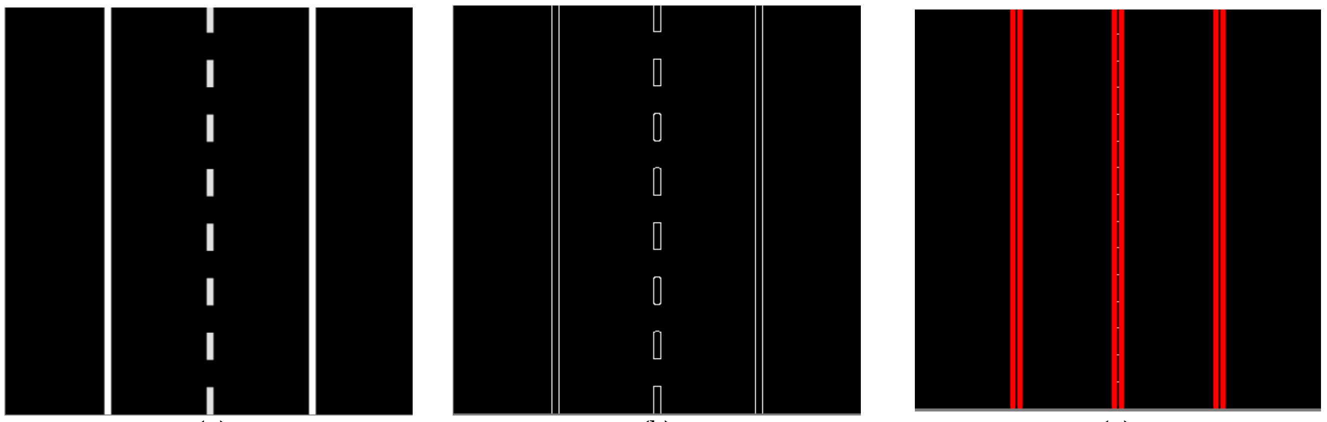
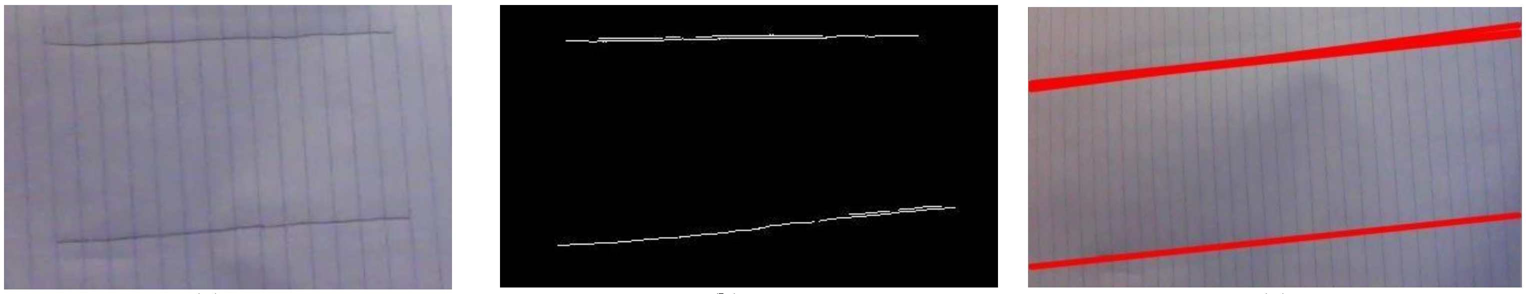


Figure 4.3: (a) Original Image, (b) Canny-Edge detection and (c) Hough Transformation

After successful testing of the functionality of Canny Detection and Hough Transformation on static image we went on to test it through the webcam of the laptop.

The frames were fetched from the live video feed of webcam and the image processing techniques were implemented on them. We made a drawing of road lanes on the paper and placed it in front of the webcam to obtain the boundaries of the lanes. Following were the results that we obtained after performing Canny detection and Hough Transformation on the live feed video.



(a) (b) (c)

Figure 4.4: (a) Sample frame from live video, (b) Canny-Edge and (c) Hough Transformation

These simulation results depicts the operation and functionality of image processing implemented on the road surface to detect the road lanes and to draw the boundaries on it. The results successfully show that we have been able to find the boundaries of the road lanes.

In Phase 3 we computed the angle required to reach the center of the detected boundaries. We encountered problems in detecting the accurate angle when we were not able to detect boundaries for parallel lanes at turns. To solve this problem we implemented a generic algorithm that can compute the correct angle even in the case of one detected lane. The figure below shows the frame with x and y axis which was used in the simulation to help us understand the angles in the image processing.

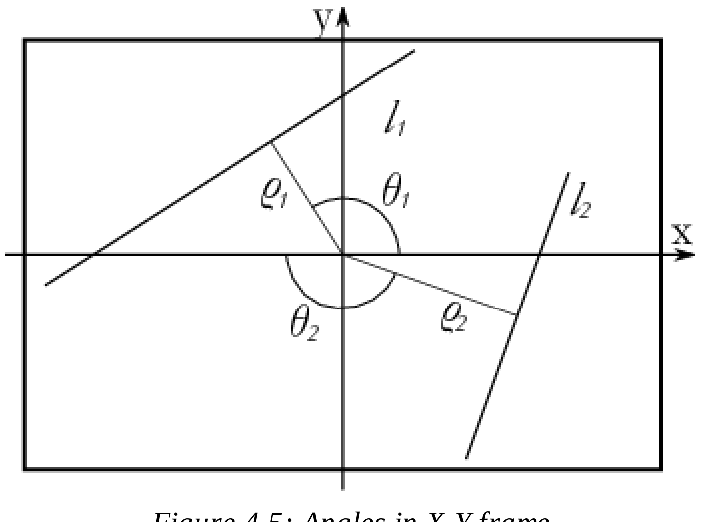


Figure 4.5: Angles in X-Y frame

The angle required to reach the origin (x=0, y=0) from the left lane l1 is ρ1 and the angle required to reach origin of the frame from the right lane l2 is ρ2. On this concept we were able to figure out the angle that we need to reach the center of the detected lane.

5. Design Implementation:

The design that was proposed in the phase 1 and phase of 2 this project was implemented and executed during the phase 3. Following are the main features of the projects that were successfully implemented.

5.1 The Big Picture

The entire envisioned algorithm is captured in the following flow chart:

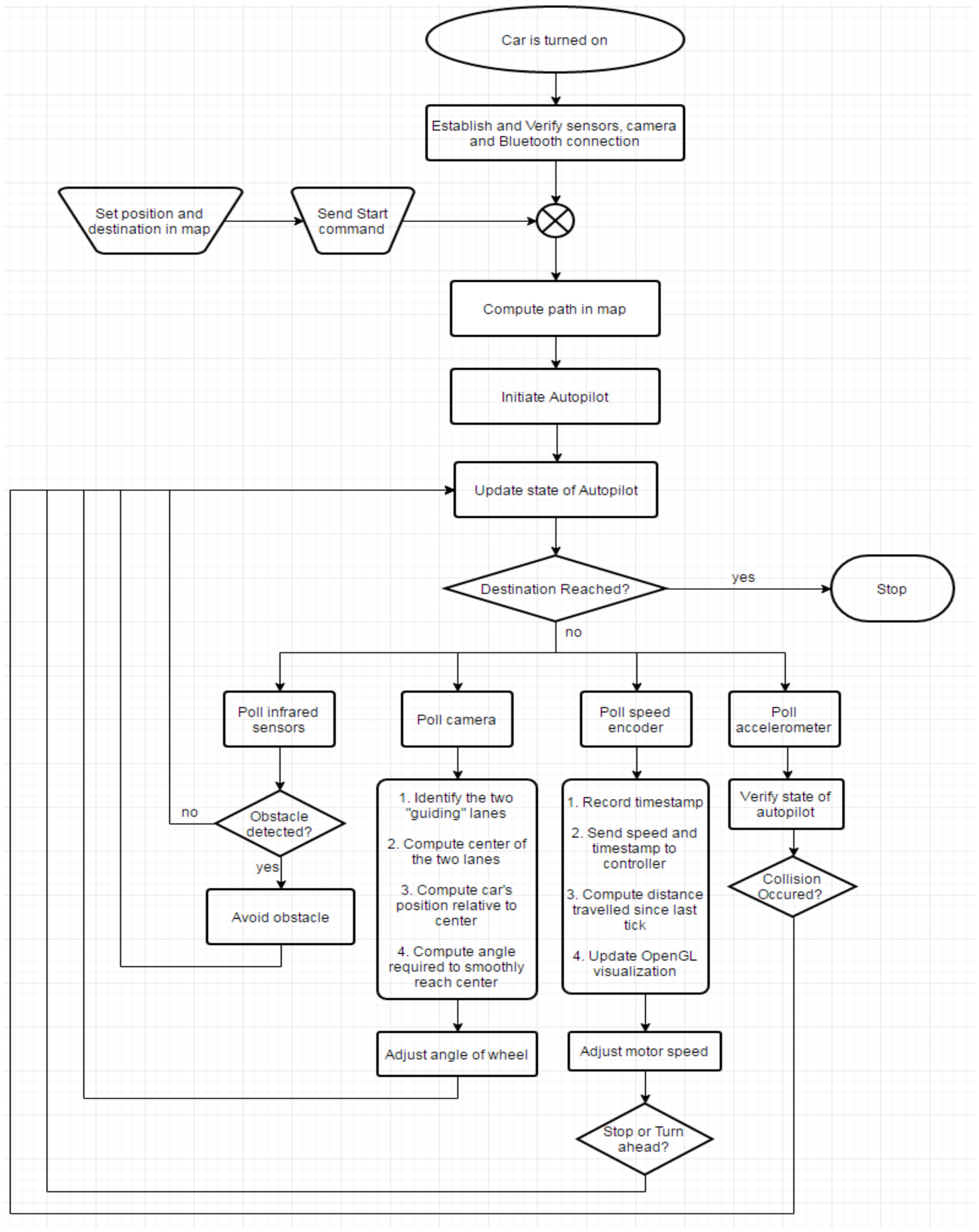

Car is turned on
Establish and Verify sensors, camera and Bluetooth connection
Set position and destination in map
Send Start command
Compute path in map
Initiate Autopilot
Update state of Autopilot
Destination Reached?
yes
Stop
no
Poll infrared sensors
Poll camera
Poll speed encoder
Poll accelerometer
no
Obstacle detected?
yes
Avoid obstacle
1. Identify the two "guiding" lanes
2. Compute center of the two lanes
3. Compute car's position relative to center
4. Compute angle required to smoothly reach center
1. Record timestamp
2. Send speed and timestamp to controller
3. Compute distance travelled since last tick
4. Update OpenGL visualization
Verify state of autopilot
Collision Occured?
Adjust angle of wheel
Adjust motor speed
Stop or Turn ahead?

Figure 5.1: Algorithm of the overall system structure

The flowchart speaks for itself and demonstrates the logical connection between the various elements in our self-driving car. Although the logic abstracts the internal implementation, it shows the feasibility of the project from a theoretical perspective and is a big stepping-stone towards development of individual components.
Some algorithms are embedded (implicitly and explicitly) into this flowchart. To elaborate on each procedure:

Detecting an obstacle:

1. Poll/Sample the Ultrasonic Sensor
2. Process the signal to translate voltage into distance
3. Reject if distance is larger than a threshold

Centering the car:

1. Process video feed to identify lanes
2. Compute position of center of lanes
3. Use proportional distance from both lanes to locate position
4. Compute angle to reach center (proportional to distance away from center)

Tracking the car:

0. Load (to-scale) model of city on computer
1. Poll the Speed Encoder every fixed period
2. Process the signal to translate voltage into speed
3. Compute distance travelled using the time since last poll
4. Send distance travelled to remote computer
5. Update position of car in the map
6. At every update check for upcoming turn using the angle between successive known positions along the path in the street map
7. At every update check if a stop is in the stop-map
8. Send alert to car if turn/stop upcoming

5.2 Autopilot

Finite State Machines are typically implemented in hardware but we have borrowed the technique for our software because it is simple yet very powerful. The following state diagram identifies the various states the autopilot can be in.

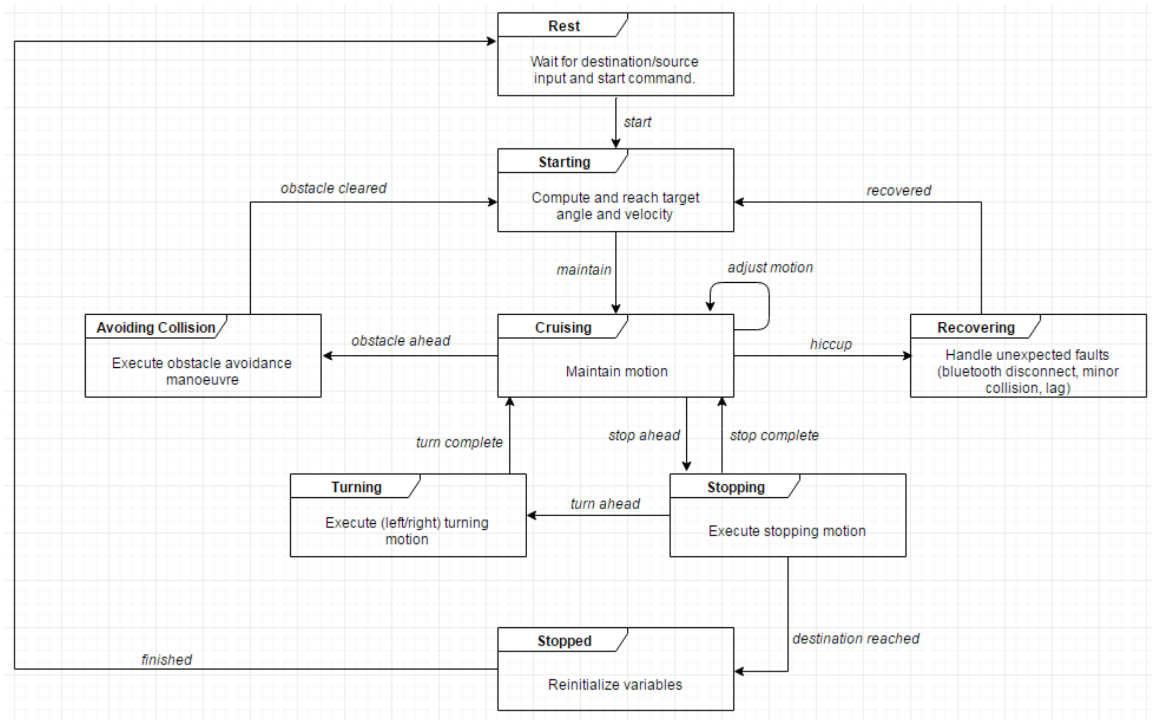


Figure 5.2: Auto-pilot state diagram

As shown, each state has an associated predetermined functionality. Essentially, the autopilot will behave differently depending on the state it is in. Furthermore, the diagram shows the actions which lead to state changes, these are known as triggers and can also have an associated function to execute before changing states.

These triggers can come from various sources: a result of polling the map and figuring out a stop or turn is ahead, a result of polling the Ultrasonic sensor and detecting an obstacle close ahead, polling the camera and determining the car is not centered, a Bluetooth disconnect or simply finishing an action.

5.3 Image Processing

The simulation that was carried out in Phase 2 and during the phase 3 we identified the boundaries of the road lanes by using different techniques. We also calculated the angle with respect to these boundaries which would enable the car to reach to the center of the lanes.

After achieving the desired simulation results we went on to implement it on our RC car by first mounting the Raspberry Camera and adjusting the camera angle to a wide view angle of the lanes. To obtain the boundaries of lanes through image processing we tapped our lanes with the white tape on our city so that it is easily visible, detectable and process able by the camera.

The flow chart below depicts the implementation of image processing that are to be used to detect boundaries and angle of the RC Car.

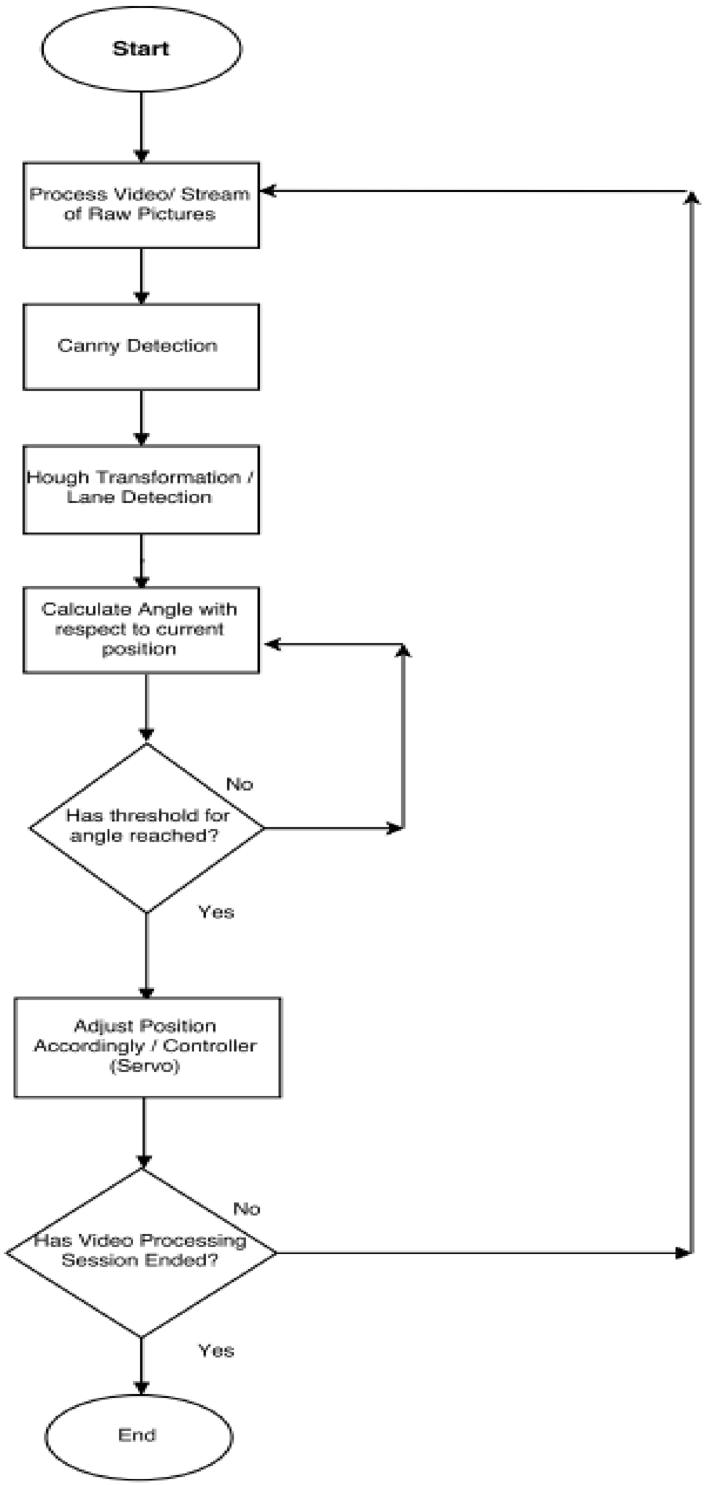
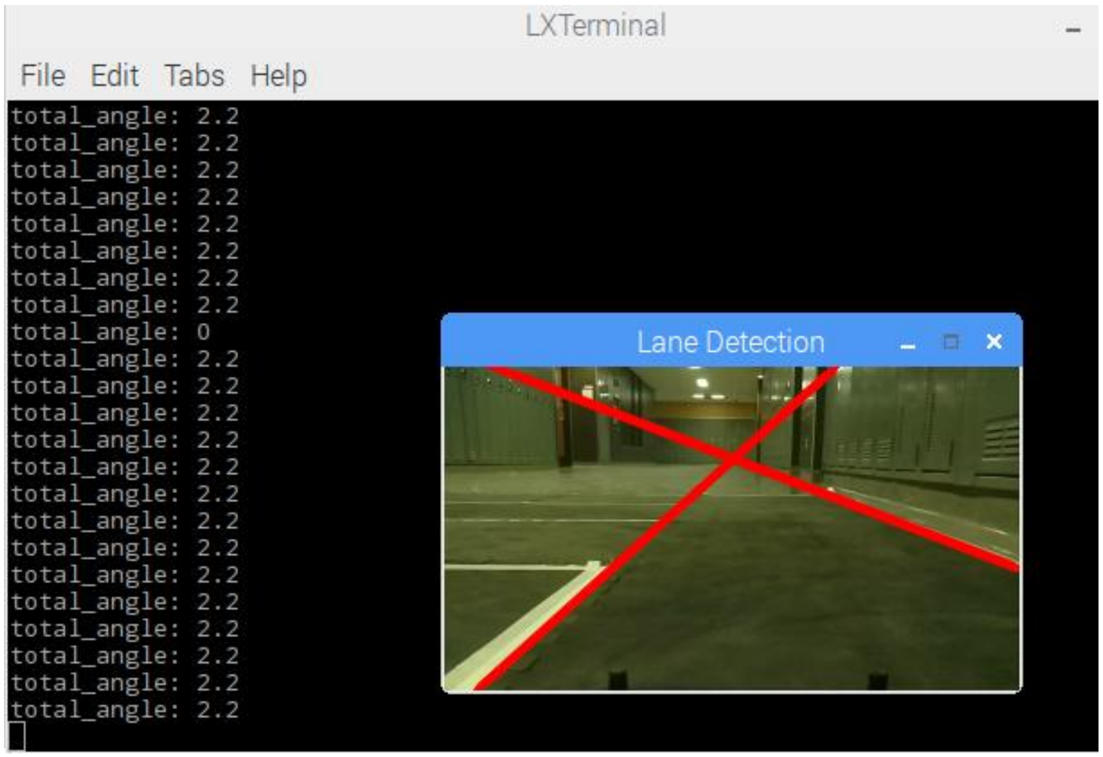

Start
Process Video/ Stream of Raw Pictures
Canny Detection
Hough Transformation / Lane Detection
Calculate Angle with respect to current position
Has threshold for angle reached?
No
Yes
Adjust Position Accordingly / Controller (Servo)
Has Video Processing Session Ended?
No
Yes
End

Figure 5.3: Flow chart of implementation of image processing

The threshold angle from the lane boundaries was adjusted in real time processing as the RC car was too close to the boundaries. The distance of lane boundaries to the RC car was recalculated and then the threshold angle was carried out by taking into account the max angle of the servo of RC car.

The threshold angle enabled us to control the car and position it to the center of the car according to our desired city lanes. After reaching the threshold the car would adjust its position through the servo and center itself with respect to the lanes.

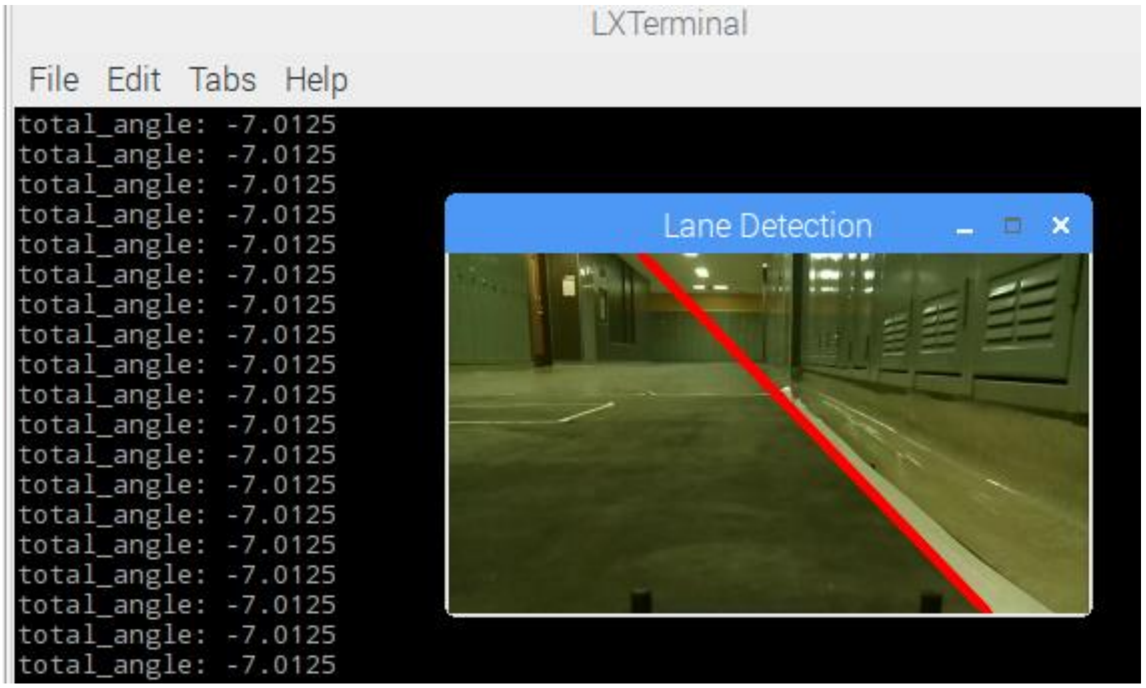
Below are the pictures that were obtained in real time implementation of the image processing.



```
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 0
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
total_angle: 2.2
```

Figure 5.4: Left Lane detection

The picture above depicts that lane boundaries are being identified by the red lines and it also shows the total angle required to reach the center of the two boundary lanes. In this case RC car is 2.2 degree away from the center of frame and the positive angle signifies that we should be turning towards the right side after reaching a threshold angle.



```
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
total_angle: -7.0125
```

Figure 5.5: Right Lane detection

This picture shows that only one lane is in the sight of the camera. The RC car in this case is very close to the right lane and this is why the angle is more than -7 degrees for the car to reach the center. Negative angle indicates that the RC should turn to the left of its current position.

5.4 Motor and Servo Control

Motor and Servo are the two main components that are critical to achieve the requirements of our project. To control these components we had to go through different techniques and tasks which are mentioned in the sections below.

5.4.1 Motor Control

The motor in the RC car is connected directly to Electronic Speed Controller and Electronic Speed Controller (ESC) has a signal connection with the Raspberry PI. Through ESC we can vary and control the speed of the motor. ESC takes the DC input voltage and simply converts it into rotating current.

ESC works on the concept of Pulse Width Modulation (PWM) which controls motor speed with a varying ratio by switching the motor on and off rapidly. The optimum frequency is required to operate the ESC so that we can provide suitable pulse widths for the speed control of the motor.

Duty cycle is also an important characteristics of PWM, it is used to describe the percentage of time a digital signal is on over a period of interval. If a duty cycle is 100% then that means the signal is high as 5V and if the duty is 0% it would be as if grounding the signal.

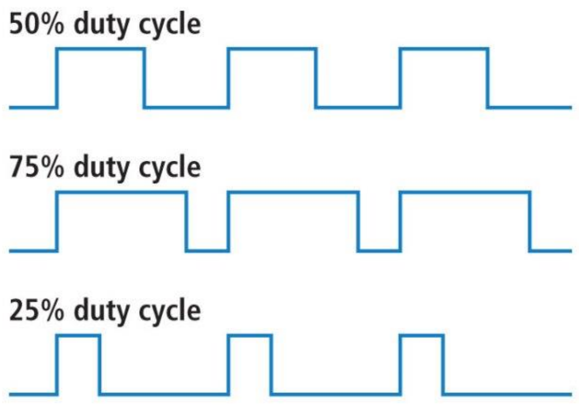


Figure 5.6: Duty cycle diagram

Hardware PWM signal pin is used on the Raspberry Pi to operate the motor through ESC. The reason of choosing the specific hardware PWM pin was to minimize the jitter of the motor and to reduce the interruption delays. We used pigpio library to implement the functionality of hardware PWM which takes frequency and duty cycle as parameters to set the speed of the motor.

The frequency required by our ESC was obtained using the formula given below:

$$\text{PWM Frequency} = \frac{19.26 * 10^6}{Clock\ rate * Range}$$

Where $19.26 * 10^6$ is the base frequency of Raspberry Pi PWM clock

Clock rate is the is the frequency at which PWM counter is incremented

Range specifies sets the limit on the frequencies that can be achieved

$$\text{PWM Frequency}= \frac{19.26*10^6}{770*50}$$

$$\text{PWM Frequency}= 500\text{Hz}$$

We found the PWM frequency to be 500Hz and using this frequency we can set the duty cycle to achieve the desired speed of the motor.

The process for speed control is sketched out in the following block diagram:

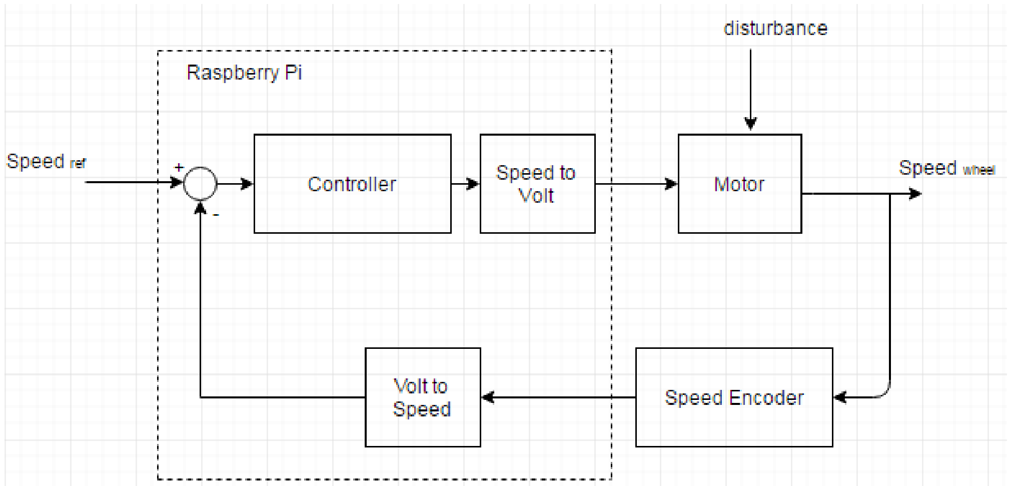


Figure 5.7: Car speed control - Feedback

The process starts by setting the reference speed through the ESC PWM signal (setting the duty cycle and frequency of ESC), translating this speed into a voltage which feeds the motor to rotate the wheels. The speed of the wheels is then sensed through a speed encoder and fed back into the system to correct any errors.

Important Notes:

- The controller incorporates the ESC and the duty cycle given to ESC then sets the speed to a certain value.
- The angle to voltage (and vice-versa) conversion and the controller (PID) will be implemented on the Raspberry Pi.
- The main disturbances anticipated here are slippage and bumps on the road.
- The reference speed will be determined by the state of the autopilot (accelerating, stopping, turning or cruising).

5.4.2 Servo Control

The servo is controlled by the microcontroller and it works on the PWM concept as well. We implemented the PID controller for the steering of the RC car. The angle obtained from the image processing and the angle from the servo is used in the PID to obtain an improved and optimum angle which will reject the deviation from the desired path.

The block diagram below shows the Steering Control System that we have implemented.

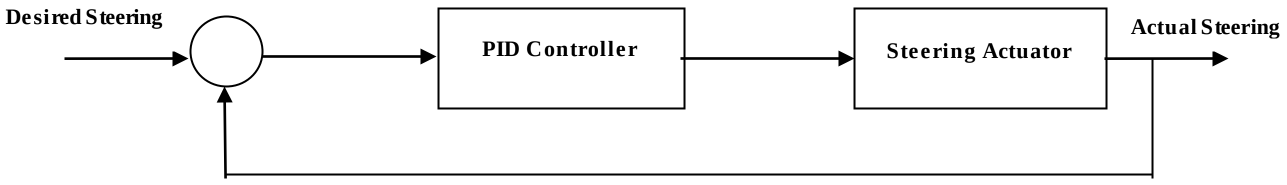


Figure 5.8: Steering control

PID coefficients plays a very important role in the stability of the steering of RC car. The coefficients for the PID control was chosen to obtain the minimum variation from the desired path. The values of Kp, Ki and Kd are given below:

Kp= 0.8 **Ki**= 0.05 **Kd**= 0.01

Steering Control System represents the PWM motor driving a steering linkage that steers the axle of RC car. The steering subsystem accepts the PWM input from the designed PID controller and outputs the current wheel angle.

5.5 Functionality of Sensors

Various sensors are being used to satisfy different objectives and provides functionalities that would be essential for our project. Below is the discussion of sensors and why are they being used.

5.5.1 Ultra-Sonic Sensor

Ultra-Sonic senor is used as a proximity sensor to detect an object using sound waves that is placed at a specific range. It measures the distance by sending out a sound wave at a certain frequency and then it listens for the sound wave to bounce back from the object. The operation can be justified with the picture given below.

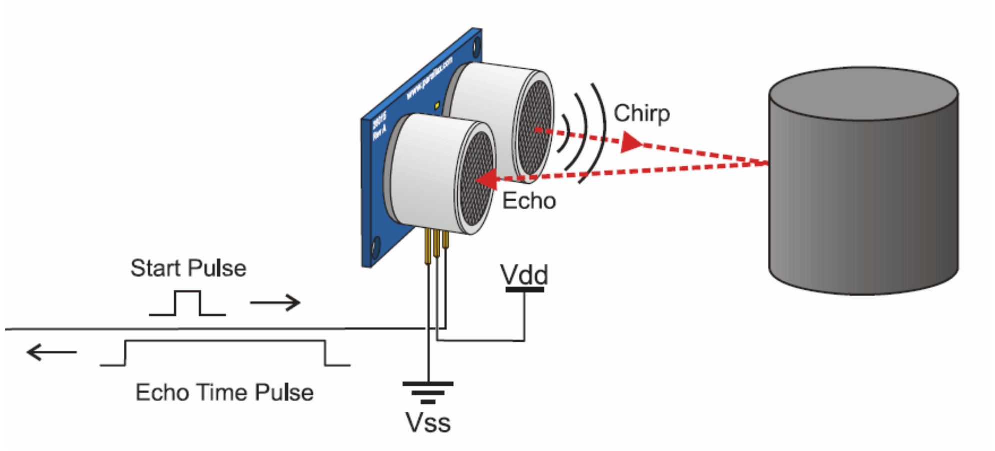


Figure 5.9: Functionality of Ultrasonic sensor

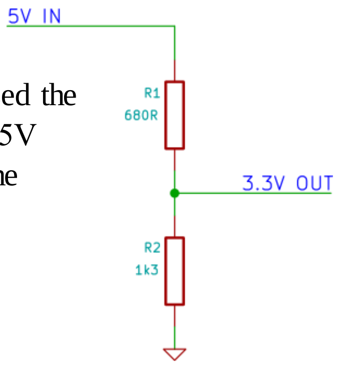
To find the distance of one round trip to the object we can use the following formula.

$$distance = \frac{speed\ of\ sound * time\ taken}{2}$$

Length of the echo pulse is proportional to the distance of the time.

The reason we are choosing ultra-sonic in our project is to detect an object at a certain range and then if RC car reaches a specific distance threshold we can stop it in front of the object using the Electronic speed controller.

To control the functionality of Ultra-Sonic sensor through the microcontroller we used the voltage divider with 1k and 2k resistor because Raspberry Pi's GPIO cannot accept 5V signal. With the help of voltage divider we decreased the voltage to 3.3V which is the minimum voltage required by GPIO pins to function properly. This is shown in the diagram.



5.5.2 Hall Effect Sensor

A Hall Effect sensor responds to the magnetic field and when the sensor is placed in a magnetic field the electrons run through from one side of the sensor to another. The sensor will then have one positive and one negative side which creates the Hall Voltage and in result it will cause the signal pin to toggle. The principle can further be shown by the figure given below

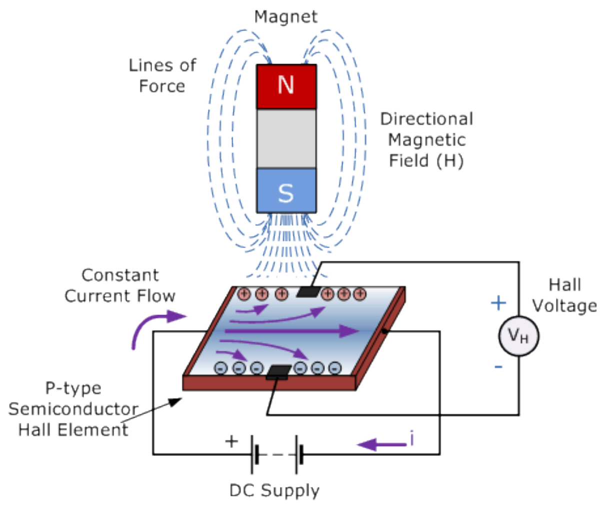


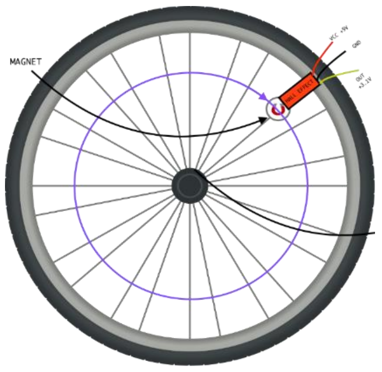
Figure 5.10: Functionality of Hall sensor

The main reason of using the Hall Effect sensor is to detect the speed and the distance RC car will cover. With the Hall sensor we can know the exact position of our car on the internal map. Whenever the RC car will move in the city it will reflect this movement to the internal map.

To achieve this goal we placed 4 magnets on the tire of the car and we also placed the sensor very close to the magnets so that the sensor can detect the magnetic field easily. The distance can be calculated using the formula given below that takes into account the circumference of the RC car tire and number of magnets being used.

$$Distance\ Travelled = \frac{Circumference\ of\ the\ tire}{4}$$

Every time the magnet on the wheel will pass the sensor we will have the exact distance travelled by the RC car. The figure on the right shows one magnet connected to the wheel and the hall sensor placed near the magnet.

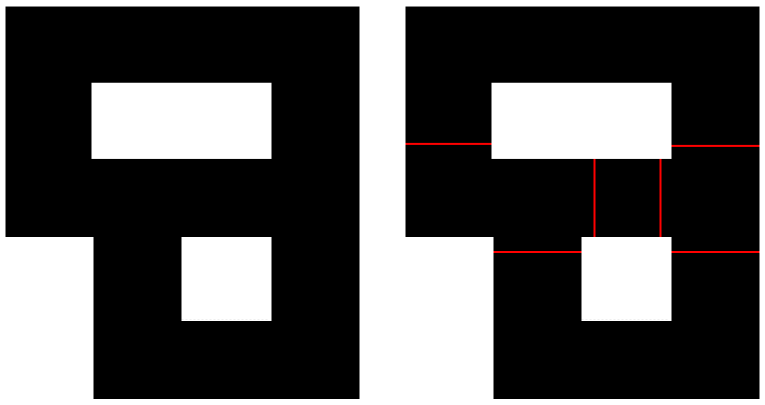


Hall Effect signal pin also requires voltage division with the 1k and 2k resistors to connect to Raspberry Pi GPIO pin in a similar way we connected the ultra-sonic sensor.

5.6 Internal Map

Since our car will be functioning inside, the internal map is our solution for recreating the GPS capabilities that a real car would have, to be able to get any destination. Another feature we embedded in it is stop signs, to relieve the image processing from this duty.

The map works by loading a scaled image of the city where the streets have been colored in a heat-map fashion to emphasize the center of the lanes and where turns occur at intersections. A similar image of the city is also loaded, where a red line is used to show a stop sign.



(a) *(b)*

Figure 5.11: (a) City streets, (b) City with stops

JPEG images are loaded from memory using an open-source external library called SOIL. Custom code reads pixel by pixel and identifies its RGB value. Pixel positions represent (x,y) coordinates on the map and the RGB values are used to score each point such that black is highest and white represents areas that are not streets. For the stop signs, the same process is used but instead we look for pure red and store those coordinates.
Now that we have a program (to-scale) representation of our city, we are able to compute paths, distances, positions of turns and stops. For visual appeal, we added models of streets where the car is able to drive and buildings elsewhere.

5.7 Path Finding

Although our model city is very small, our algorithm for internally mapping a city from a picture is generic and applicable to large cities as well. In general, to figure out how to get from point A to point B you need to figure out which turns to take, so this is why we need a path finding algorithm.

We used an external open-source library for path finding which uses a search algorithm called AStar. This is a popular algorithm used in video games to find the minimal path between two points using a heuristic function (ex: Euclidian distance).

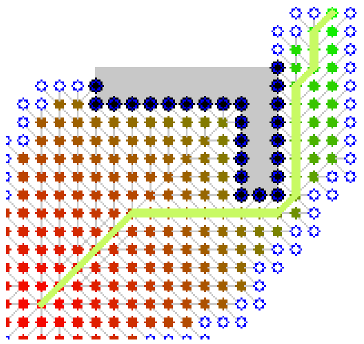


Figure 5.12: Visual representation of path finding algorithm [3]

Since we already mapped our city into coordinates of areas we can drive on and areas we cannot. We simply feed the algorithm the boundaries of the city and the areas we cannot drive at, the source and destination of the car and a path (sequence of coordinates) is generated.

The path will be used to compute turns so the image processing is relieved from this duty and allows the car to reach its destination efficiently. Otherwise, the car would wander blindly.

5.8 Software Implementation

A domain model or class diagram would be helpful to showcase our software architecture but this is still in the preliminary stages of design.

For now, we have identified the following classes in our domain model:

<u>Car side</u>:

Class Name	**Purpose**
Autopilot	Track the states Manage the other classes
BluetoothManager	Exchange data with computer
Driver	Control the motor and the wheel
Timer	Keeping track of elapsed time
Camera	Process video feed Determining angle needed to maintain alignment
UltrasonicSensor	Process ultrasonic sensor signal Determine if obstacle
Accelerometer	Process accelerometer signal Verify state of car Determine if collision

SpeedEncoder	Process speed encoder signal Compute speed of car Compute distance
Controller	Compute new angle based on reference and sensing data
VoltageConverter	Convert speed/angle to voltage and vice-versa

Table 5.1: Domain Classes at Car side

Remote side:

Class Name	**Purpose**
Renderer	Visually represent the car in its environment
InternalMap	Hold data regarding streets layout Hold data regarding stops
BluetoothManager	Exchange data with car
Autopilot	Compute path to reach destination Manage the other classes Update position of car

Table 5.2: Domain Classes at Remote side

5.8.1 Ultra-Sonic Sensor

Trigger and Echo pulses are being used in the code to compute the distance of an obstacle placed a certain range.

Following function in our code returns the distance in cm depending on the trigger response of the ultrasonic sensor.

```
int getCM() {
        //Send trig pulse
        digitalWrite(TRIG, HIGH);
        delayMicroseconds(20);
        digitalWrite(TRIG, LOW);

        int attempts = 0;
        //Wait for echo start
        while(digitalRead(ECHO) == LOW) {
            if (++attempts >= MAX_ATTEMPS) {
                break;
            }
        }

        //Wait for echo end
        long startTime = micros();
        while(digitalRead(ECHO) == HIGH);
        long travelTime = micros() - startTime;

        //Get distance in cm
        int distance = travelTime / 58;

        return distance;
}
```

This code explains the process of triggering and echo for the ultrasonic sensor. First the sensor is triggered by a high pulse for 20 microseconds and then we give a low pulse beforehand to ensure a high clean pulse. This pulse then listens and waits for a response from the echo pulse.

When the echo pulse senses an object it lowers the echo line and sets it to LOW state. Once this echo pulse stops detecting an object it changes its echo line to HIGH state and then we compute the distance it takes for the echo line to hit the object. The echo line is simply a pulse whose width is proportional to distance of the object.

Before the echo pulse is set to HIGH state we start the time in microseconds and once it is set to HIGH state we compute the total travel time it takes for the echo line to hit the object. Then the distance is calculated in cm by just dividing the total travelled time (microseconds) with 58. The reason we divided it with 58 is because 1 cm is equal to $\frac{1\,\mu s}{58}$.

If this distance found is less than the obstacle threshold distance then that means an obstacle has been detected.

5.8.2 Speed Encoder (Hall Sensor)

We are using the Hall sensor as our speed encoder so that we can monitor the real time location of our RC car on our city. We achieved this by using the main features of speed encoder such speed and distance. Following code was used to calculate the distance, speed and displacement of the RC car.

```
void SpeedEncoder::InterruptHandler() {

    long start_time= micros();
    long elapsedtime;
    long interrupttime;

    if(0 == digitalRead(SIG)){

        interrupttime = micros();
        rotation = rotation +1;
        distance = rotation * WHEEL_CIRCUMFERENCE;
        displacement += distance;
        elapsedtime = interrupttime - start_time;

        speed = distance/elapsedtime;

        start_time = interrupttime;

        printf("Magnet Detected!\n");

    }

    else if(1 == digitalRead(SIG)) {
        delay(10);
        if(1 == digitalRead(SIG)){
            while(!digitalRead(SIG));
            printf("Nothing...\n");
        }
    }
    cout << "Total Distance is: " << distance;
    cout << "Speed is: " << speed * 1000000; //cm/s
    cout << "Displacement: " << displacement;
```

This function starts by computing the start time in microseconds. Once the magnet is detected by the hall sensor the signal goes to the 0 state. We then take into account the interrupt time as soon as the hall sensor detects magnet. The rotation is then incremented by one and then the distance is computed multiplying the rotation with the circumference of the wheel. The reason we are multiplying the number of rotations with wheel circumference is because every time the magnet on the wheel will pass the sensor we will have the exact distance travelled by the RC car. Then we went on to find the displacement and the elapsed time, elapsed time is the time difference the speed encoder detected the magnet to the beginning of the time.

Once we have the elapsed time and distance we can compute the speed of the RC car by dividing the total distance with the elapsed time. At the end we initialized the start time to the interrupt time because we want the updated time for the next detected magnet. To get the speed in cm/s rather than in cm/µs we multiplied the speed with 1000,000.

It is worth mentioning that we are using interrupt handling for the speed encoder with the help of wiring pi library. Following is the function we used for an interrupt to occur on our signal pin of hall sensor whenever it detects a magnet.

```
void SpeedEncoder::SetInterrupt() {

    //generate an interrupt on high-to-low transitions on gpio SIG
    wiringPiISR(SIG, INT_EDGE_FALLING, &SpeedEncoder::InterruptHandler);
}
```

The above function will be called when the interrupt triggers and in result of the interrupt it will call the InterruptHandler function that is shown at the beginning of this section. We are using the interrupt because we don't want to miss this high priority task which is essential for us to locate the position of RC car in our city and then reflect this location through speed encoder in the internal map which has been made in OpenGL.

5.8.3 Image Processing

There were different scenarios that we encountered while programming for image processing. First we found the left and right lanes based on where the left and right intercepts are located with respect to the center of the frame. Following code was used to find either we found a left lane or right lane in our city using image processing techniques.

```
if (angle != 0) {

    Vec2f cart_line;
    cart_line[0] = (-cos(theta) / sin(theta));
    cart_line[1] = rho / sin(theta);

    Point intercept;
    intercept.y = frame_center.y;
    intercept.x = (frame_center.y - cart_line[1]) / cart_line[0];
    side intercept_side = center;

    if (intercept.x > frame_center.x) {
        intercept_side = right;
    }
    else if (intercept.x < frame_center.x) {
        intercept_side = left;
    }
```

This code first defined both x and y intercepts based on the center of the frame. After defining the intercepts the x intercept was compared with the frame center to find out either it's a right lane or left lane that has been detected.

After we found out the lanes we went on to find the closet intercept for both cases so that we can get the angle using those closest intercepts. To find out the angle required to reach the center of the frame from either right or left side of the lane we used the following formulas for each case:

<u>Angle required from left lane:</u>

$$\theta = a * x\ intercept$$

Where θ is the maximum angle RC car can achieve

X intercept is the closest intercept from the left lane

In case of θ being equal to 22 degrees we can find out a

$$a = \frac{22}{frame\ center}$$

So the total angle required would be

$$\theta = \left(\frac{22}{frame\ center}\right) * x\ intercept$$

<u>Angle required from right lane:</u>

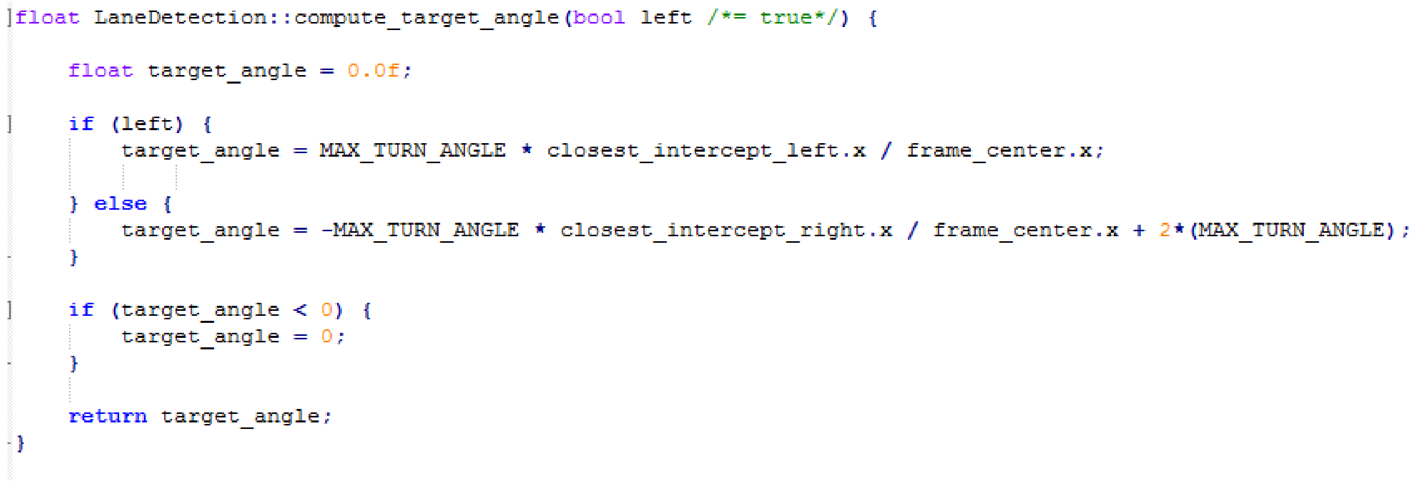
$$\theta = a * x\ intercept + 2(Max\ turn\ angle)$$

$$a = -\frac{22}{frame\ center}$$

So the total angle required would be

$$\theta = \left(-\frac{22}{frame\ center}\right) * x\ intercept + 2 * (Max\ turn\ angle)$$

The above formulas were applied in the following code to obtain the target angles:

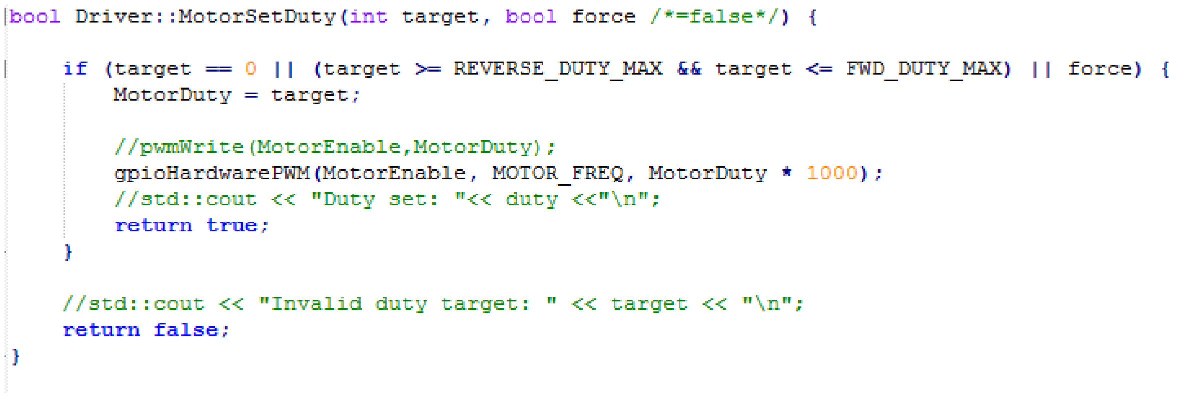


```
float LaneDetection::compute_target_angle(bool left /*= true*/) {

    float target_angle = 0.0f;

    if (left) {
        target_angle = MAX_TURN_ANGLE * closest_intercept_left.x / frame_center.x;

    } else {
        target_angle = -MAX_TURN_ANGLE * closest_intercept_right.x / frame_center.x + 2*(MAX_TURN_ANGLE);
    }

    if (target_angle < 0) {
        target_angle = 0;
    }

    return target_angle;
}
```

5.8.4 Motor and Servo

Motor:

Hardware PWM signal pin is used on the Raspberry Pi to operate the motor through ESC. The reason of choosing the specific hardware PWM pin was to minimize the jitter of the motor and to reduce the interruption delays. We used pigpio library to implement the functionality of hardware PWM which takes frequency and duty cycle as parameters to set the speed of the motor. The code shown below can be used to set the target speed for the motor.

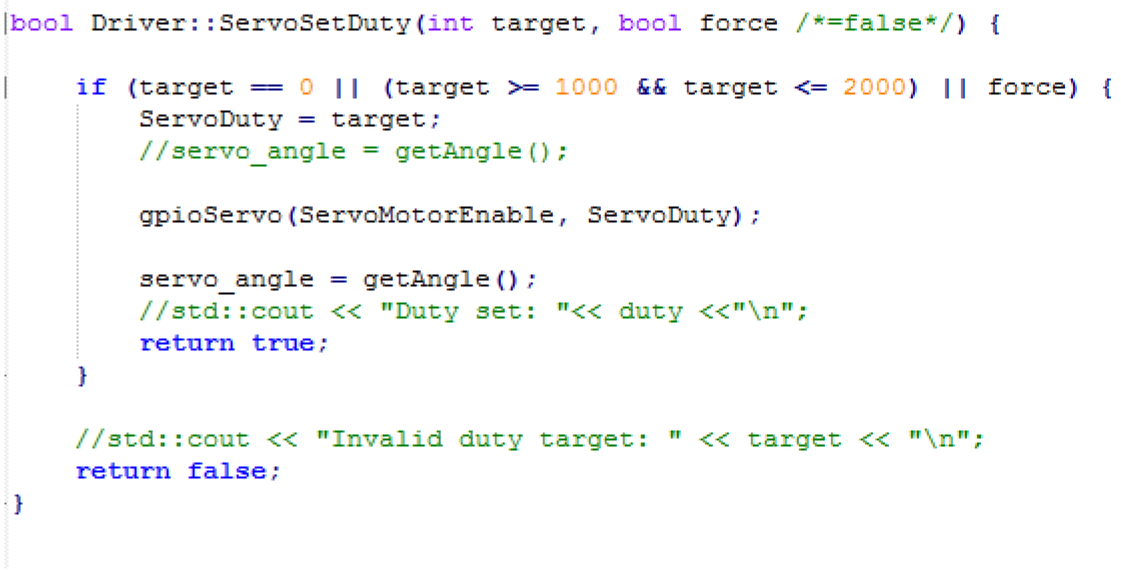


```
bool Driver::MotorSetDuty(int target, bool force /*=false*/) {

    if (target == 0 || (target >= REVERSE_DUTY_MAX && target <= FWD_DUTY_MAX) || force) {
        MotorDuty = target;

        //pwmWrite(MotorEnable,MotorDuty);
        gpioHardwarePWM(MotorEnable, MOTOR_FREQ, MotorDuty * 1000);
        //std::cout << "Duty set: "<< duty <<"\n";
        return true;
    }

    //std::cout << "Invalid duty target: " << target << "\n";
    return false;
}
```

The above function can be used to set a specific speed value that we wish to attain for the motor. Using the pigpio library we wrote on the Hardware PWM GPIO which takes into account the motor frequency and the duty.

Servo

Servo functions on the concept of PWM as well. Duty Cycles are specified to turn the servo according to what we acquire. Servo angle can be carried out by mapping the servo duty in the range of 1000 to 2000. By calculating this servo angle we would be able use it in our PID controller. Following code can be used to set the servo duty and then calculating the servo angle according to the duty.



```
bool Driver::ServoSetDuty(int target, bool force /*=false*/) {

    if (target == 0 || (target >= 1000 && target <= 2000) || force) {
        ServoDuty = target;
        //servo_angle = getAngle();

        gpioServo(ServoMotorEnable, ServoDuty);

        servo_angle = getAngle();
        //std::cout << "Duty set: "<< duty <<"\n";
        return true;
    }

    //std::cout << "Invalid duty target: " << target << "\n";
    return false;
}
```

5.8.5 Multithreading

Different threads are being used simultaneously to perform various operations. The interrupts are also taking place while these threads are executing. Some of the threads that we are using in our project are given as below:

- Lane Detection
- Speed Detection
- Obstacle Detection
- Auto-Pilot
- TCP/IP Connection

Image processing is in continuous use to detect the boundaries of the lanes and compute the angle accordingly. This angle is then given to Driver class who take cares of the steering.

The speed encoder has to function all the time as well to compute the distance RC Car has travelled so that this can be reflected in our internal map. A small variation in distance can cause the RC car to deviate from its path.

The detection capability must in the functioning state to encounter any obstacle in the path of the RC Car. This would allow the RC car to stop at a specified distance before an obstacle.

Auto-Pilot is one of the most important feature in our project that must be in the execution state all the time. With the help of Auto-Pilot we can go into different states of the RC Car and control all the other threads that have been mentioned above.

The connection between Raspberry and computer must be established without any interruption so that we can send and receive different messages to and from the Raspberry Pi. This thread is critical in the exchange of different protocols that have been set up in Connection Manager class.

5.8.6 Real Time Processing (*in progress)

Periodic Tasks		
Polling camera / sensors / speed encoder		
Update state of autopilot		
Video processing		
Calculating speed		
Adjusting motor speed		
Adjusting servo		
Aperiodic Tasks		
Obstacle avoidance/detection		
Connectivity between PC and RPi		

Table 5.3: Real-Time Scheduling

5.8.7 Libraries and Config file

<u>Libraries used:</u>

We used couple of libraries in our project to achieve different tasks. Following is the table of the libraries that were used and what was the reason of using those libraries.

Library	Purpose
Standard Template Library (STL)	To implement data structures and algorithms
Open CV	To implement Image Processing techniques
Wiring Pi	GPIO access library for Raspberry PI
PID	To increase the stability in the steering of RC Car

Table 5.5: List of libraries used

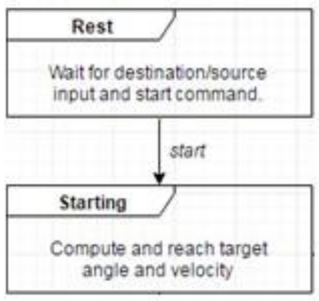
<u>Config file:</u>

The config file served the purpose of declaring all the parameters that were to initialize multiple times for the testing. These variables were changed with the help of this file without getting into the hassle of compiling the code again and again. Following are the variables that we declared in this file:

```
OBSTACLE_THRESHOLD:10
CONSTANT_FORWARD_SPEED:10
CONSTANT_BACKWARD_SPEED:1
WHEEL_RADIUS:2.65
PORT_NUM:51717
MAX_DELAY:500
MotorPin1:27
MotorPin2:22
MotorEnable:13
ServoMotorPin1:10
ServoMotorPin2:9
ServoMotorEnable:12
TRIG:15
ECHO:4
MAX_ATTEMPS:10000
MAX_RANGE:50
SIG:17
MAX_ANGLE:22
MAX_TURN_ANGLE:22
LOOP_INTERVAL_TIME:0.1
Kp:0.1
Ki:0.01
Kd:0.5
ADJUST_ANGLE_SENSITIVITY:5
TURN_DIST_THRESHOLD:5
TURN_SPEED_THRESHOLD:1
STOP_DIST_THRESHOLD:15
```

5.9 Finite State Machine

The purpose of the finite state machine (fsm) is to programmatically represent the possible states and corresponding actions of the car. To represent our states, triggers and actions in a state machine we used an open-source project called CppFSM.
Let's take these two states as an example:


Rest
Wait for destination/source input and start command.
start
Starting
Compute and reach target angle and velocity

First we declare the states and triggers: enum States { REST, STARTING}; enum class Triggers { start };

Then we declare the FSM:
FSM::Fsm<States, States::REST, Triggers> fsm;

Finally we declare the transitions:

```
fsm.add_transitions({{

States::REST, //from state

States::Starting, //to state
[&]{return ready_to_start; }, // condition

[&]{start_autopilot(); } }, // action

});
```

That is all, now a trigger can be called:
Fsm.execute(Triggers::start);

This will check the condition we set (ready_to_start), if it is true, then it will change the state of the FSM and execute the action we had set. At any time we can check the state of the FSM using fsm.state(). The program can now execute different code depending on its state.
Any arbitrarily large state diagram can be implemented with this method by simply declaring the states, triggers, transitions and actions.

6. Schematic

Below is the Electrical Schematic diagram that has been used to assemble the hardware components of the RC Car.

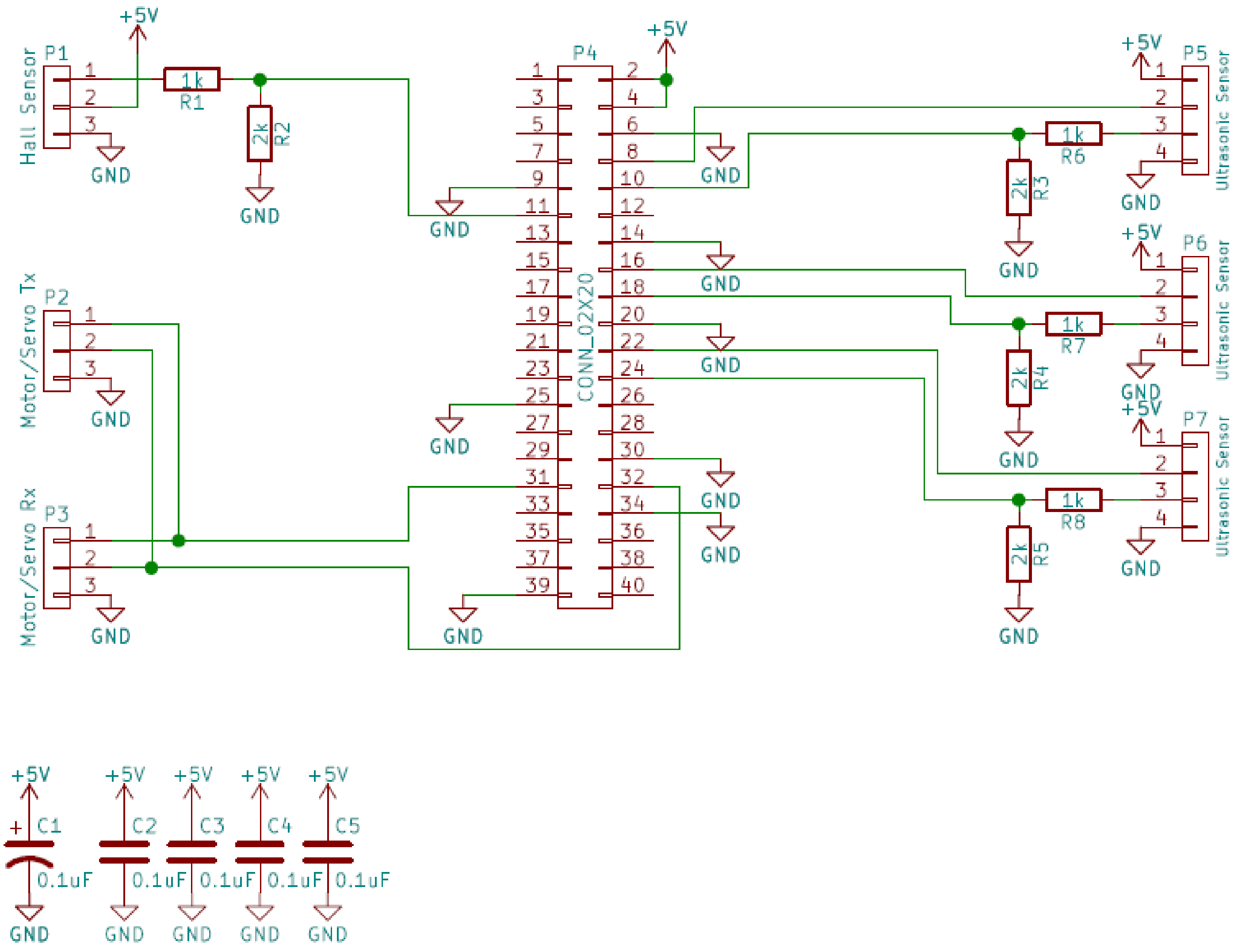
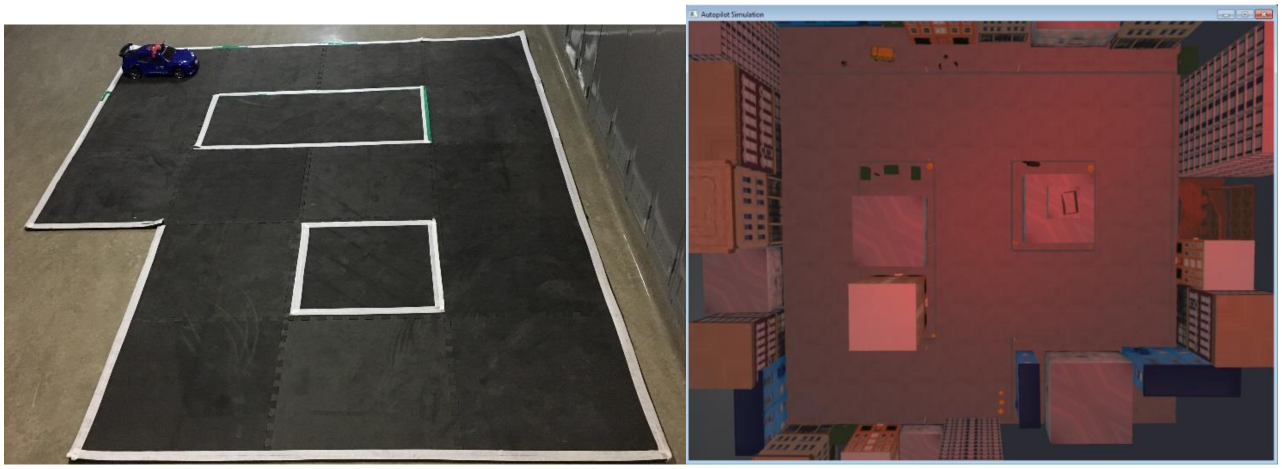


Figure: 6.1 Overall Schematic diagram

7. RC Car and City

<u>City Specifications:</u>

- Lanes are 63.5 cm wide
- The white boxes signifies buildings
- Dimension for the city is 254 cm in width and 317 cm in length. City uses 15 foams pieces and each foam puzzle has a dimension of 63.5 cm.
- Obstacles are not shown in the city



(a) *(b)*

Figure 7.1: (a) Actual City and (b) Simulated City in OpenGL

<u>Car Specifications:</u>

1/10 scale RC car

- Operational Voltage(V) : 7.2 V
- Run Time: 15 minutes
- Dimensions:
  - Length: 36 cm
  - Width: 22 cm
  - Height: 11.2cm
  - Ground Clearance: 0.45cm

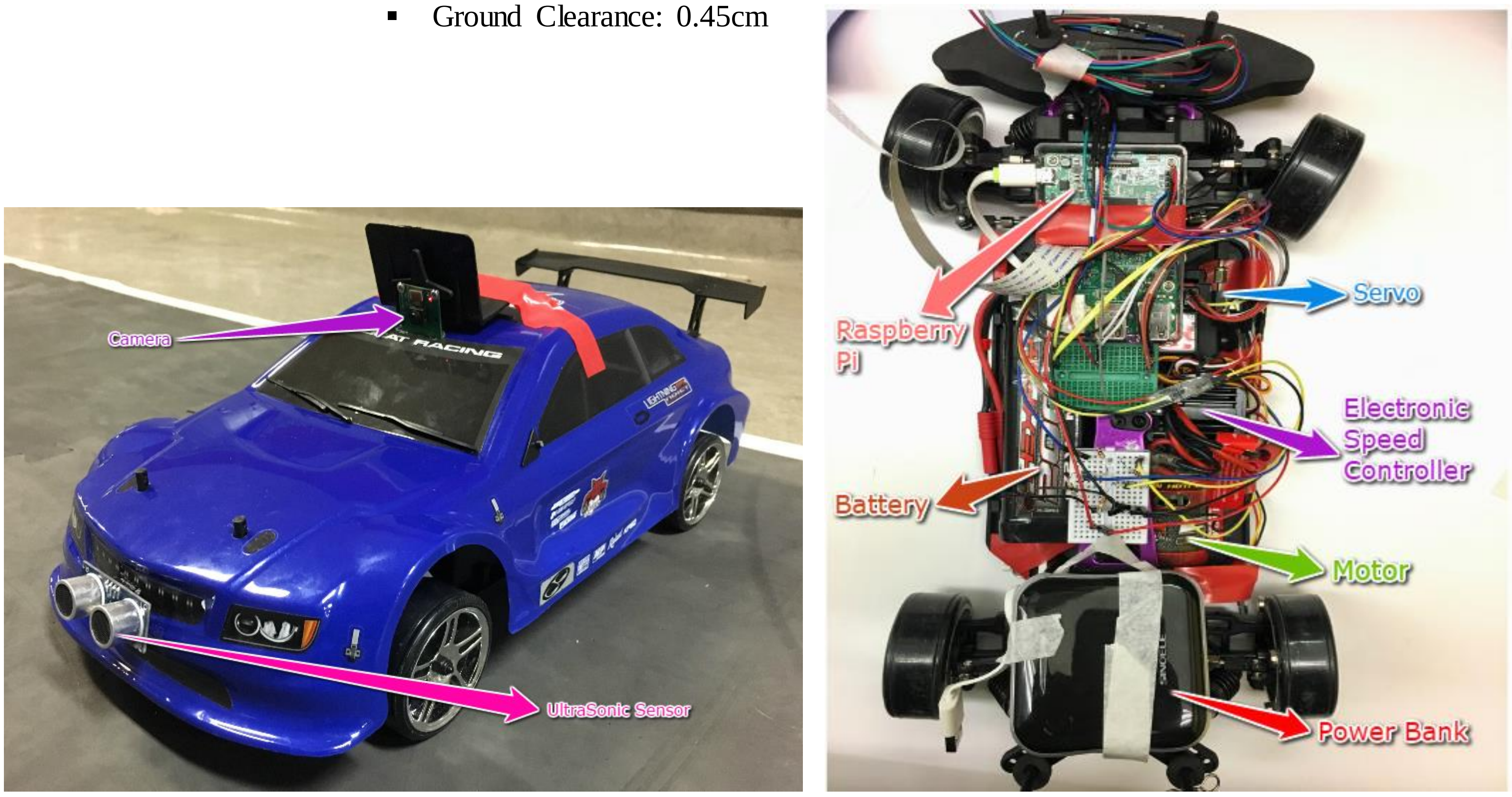


Figure 7.2: Assembly of RC car

8. Testing and Results

In this section, we first test the individual components, then create their respective code on C++ and then finally assembled the car and tested against our requirements.

8.1 Ultrasonic sensor

In order to test the ultrasonic sensor the following circuit was used:

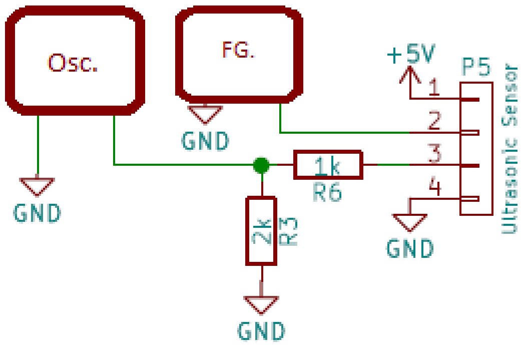
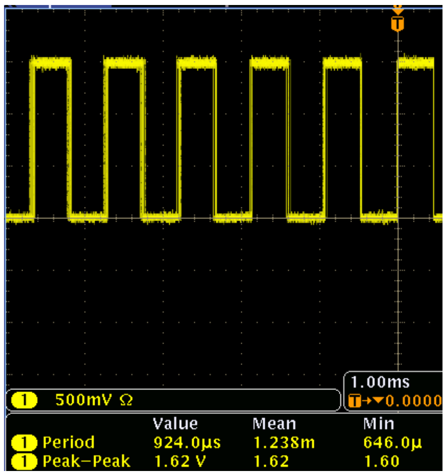


Figure 8.1: Test circuit of Ultrasonic sensor

Function Generator was set to Square wave with 40kHz input

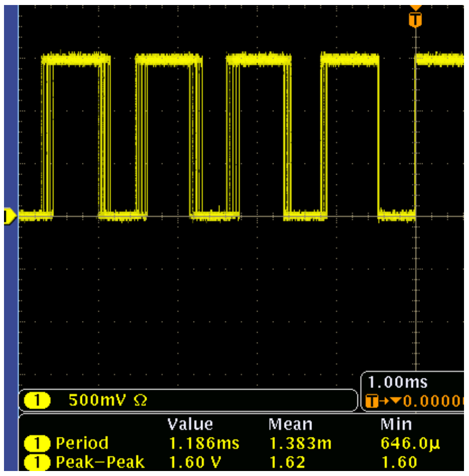
- Pin 1 = Power Supply
- Pin 2 = Trigger
- Pin 3 = Echo
- Pin 4 = Ground

Then using the oscilloscope at ECHO pin, the following plots were obtained:



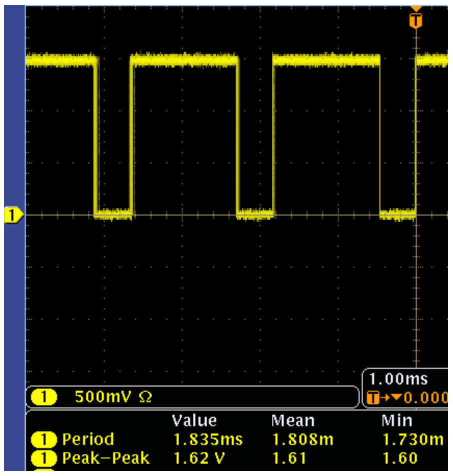
```
1  500mV Ω                 1.00ms
                           T→▼0.0000
               Value      Mean      Min
1 Period       924.0µs    1.238m    646.0µ
1 Peak-Peak    1.62 V     1.62      1.60
```

(a)



```
1  500mV Ω                 1.00ms
                           T→▼0.0000
               Value      Mean      Min
1 Period       1.186ms    1.383m    646.0µ
1 Peak-Peak    1.60 V     1.62      1.60
```

(b)



```
1  500mV Ω                 1.00ms
                           T→▼0.000
               Value      Mean      Min
1 Period       1.835ms    1.808m    1.730m
1 Peak-Peak    1.62 V     1.61      1.60
```

(c)

Figure 8.2 ECHO output at (a) 5cm, (b) 10cm and (c) 20cm

Distance / cm	Time Period / ms
5	0.924
10	1.186
20	1.835

Table 8.1: Distance – Time period relation

It can be noted that further the distance, larger the pulse width. Hence this result is in agreement with our theory.

8.2 Hall sensor

The hall sensor was connect in the following manner;

- Pin 1 = Signal (Raspberry GPIO pin)
- Pin 2 = Power Supply
- Pin 3 = Ground

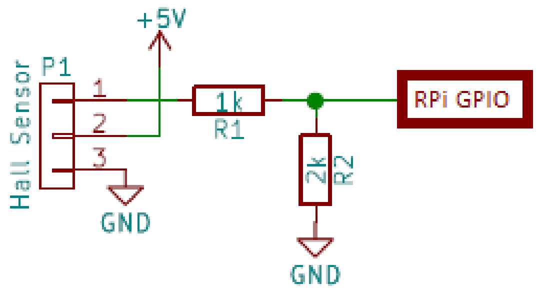


Figure 8.3: Test circuit of Hall sensor

Next a magnet was brought in the vicinity of the sensor. The LED of the sensor blinks when the magnetic field is detected.

It was noted that the maximum detection range was up to 5cm.

8.3 Motor and Servo

We programmed the motor using hardware PWM of the micro-controller. In C++ a program was created that first assigns motor pin to corresponding micro-controller's pin and then the duty cycle of the motor was varied such that it rotates clockwise (go forward) or counter clockwise (go backward) correspondingly.

The duty cycle scale of our program for the motor is summarized in the following table;

Duty Cycle / %	**Direction of rotation**
100 to 51	Clockwise – speed decreasing with lower duty cycle
50	Stop
49 to 1	Counter Clockwise – speed increasing with lower duty cycle
0	Stop

Table 8.2: Motor – Duty cycle and Direction of rotation

Using the similar process we made duty cycle scale of servo motor as shown in the following table;

Duty Cycle / %	**Direction of rotation**
20	Set full right
15	Set straight
10	Set full left

Table 8.3: Servo– Duty cycle and Direction of rotation

8.4 Test Cases

After testing each component individually we combined everything to make the complete system and tested it against our requirements. The following table shows the results.

	Acceptance Criteria	**Result**	**Pass / Fail / Comment**
Speed test	Speed should not exceed 1.2 m/s	Speed obtained was 0.135 m/s	Pass
Obstacle detection range	Detect an obstacle at least 50cm away	Detected an obstacle 65cm away from object	Pass
Stoppage of car before an obstacle	Stop before an object that appear 10cm in front of it	Stopped 15cm away from the obstacle	Pass
Travelling distance (Battery life)	It can remain in motion for 20 minutes	Run time was 32 minutes	Pass
Weight of the payload and car	Total weight of the car should not exceed 2.65 kg	Weight of the car came out to be 900g	Pass

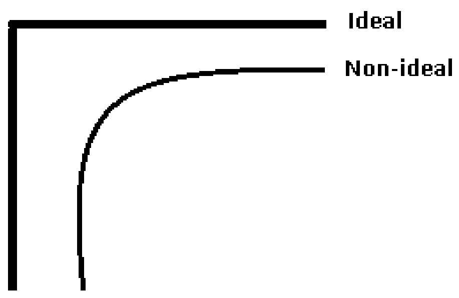
Table 8.4: Test cases

9. Project Limitations

1) Image Processing

Due to time constraint we had limit our scope and hence our lane detection algorithm was made such that it detects straight and sharp turn lanes only; curved lanes are not detected. Furthermore, since the micro-controller has a limited memory available, video processing cannot be done at resolutions more than 640x480p without experiencing delays.

2) Displacement/Speed accuracy



Internal Map is used as an alternative to GPS so that we could demonstrate our project indoors. Distance travelled in our internal map is based on number of rotations of the wheel of the car. In the case when the car is in turning state, the car follows the "non-ideal" trajectory and since the calculation was done based on the "ideal" trajectory, our system is inaccurate. And since speed calculation was based on distance measurement, our speed is also inaccurate with the same ratio as distance.

3) Connection between micro-controller and computer

A wireless internet connection is required to connect the micro-controller and remote computer. In general as it is known wireless networking signals are subject to a wide variety of interference and since data processing is done remotely on a computer and results transferred to the micro-controller via wireless internet, delays exists in our system. Moreover, it is important that the micro-controller and the computer are connected to a common wireless internet connection.

10. Project Planning

Scrum is used as a project management tool which is used in Elec 390. Every Product Backlog Item is a unit of work that needs to be accomplished, can be broken down into many subtasks and are classified according to its type, either an feature or knowledge acquisition (research). The story title is the title of the task and it's classified according to its priority in the project. The conversation gives u better understanding and detailed description of the product backlog item. Lastly, the confirmation is list of objectives that needs to be met to complete the product backlog item.

10.1 Product Backlog

Build an autonomous RC car efficiently at a low cost scale. Need to understand what hardware and software that is going to be needed, as well to understand the amount of time required to build this project and to estimate the cost of the entire plan. If there are alternatives ways to plan the project, then the team needs to find the most efficient and cost effective alternative.

Type: Knowledge Acquisition	**Story Title:** Autopilot	**Priority:** 1
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I need to understand how the autopilot is implemented on the RC car so that the RC car be autonomous.	The autopilot for RC cars requires many different kinds of sensors all working together. Need to understand the hardware and the software used to implement the Autopilot. Different kind of sensors need to be overlooked.	The car should be able to drive autonomously in the small scaled city by us.

Type: Knowledge Acquisition	**Story Title:** RC car & City Research	**Priority:** 2
Product Backlog Item	**Conversation**	**Confirmation**
As an electronics specialist, I need to understand how the RC car will either be built or modified with the sensors so that the autopilot can be implemented to a RC car in our own built city.	I will need to understand and research on how different kinds of electronics parts can coexist with the RC car as a system and to understand how the RC car functions electronically. Need to understand the power requirements of the sensors and the RC car. I also need to understand what type of materials will be necessary to build the city. I will also need to make a budget for both the RC car and city and research on the best way to implement those together.	I should have enough knowledge on how the RC car should function with the other sensors and electronics. Research on the materials and size needed to build the city.

Type: Building	**Story Title:** Building RC car & City	**Priority:** 3
Product Backlog Item	**Conversation**	**Confirmation**
As an electronics specialist, I need to either build or modify the RC car and design a city for the RC car so that the autopilot can be implemented to an RC car.	The RC car should be compatible to the type of inputs I'm going to use for autopilot. The city is going to be designed according to the scale of the RC car. The RC car will have autopilot and its going be perform such task in the city.	The RC car should function with the other sensors and electronics. The city will need to accommodate the RC scale. Hence, the dimensions of the lanes will be designed according to the dimensions of the RC car.

Type: Simulation	**Story Title:** Autopilot Simulation	**Priority:** 4
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I need to implement an interface that does simulation of the autopilot so that the autopilot can be tested virtually through a software.	The simulation of the autopilot is program that simulates every feature listed in the product backlog item. The simulation is designed to demonstrate how the autopilot will be functioning and it will be used to control the RC car in the mini scaled city. The simulation consists of the exact replica of the mini-scaled city.	The autopilot simulation should consist of simulation every feature listed in the product backlog item.

Type: Feature	**Story Title:** Line Alignment	**Priority:** 5
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I want to make sure that the RC car always remain in between two lanes so that it follows traffic rules and it drives properly.	For the RC car to be able to drive between two lanes at all time, I will have to use, program and understand sensors that can help the car detect and follow the white lines in between the lanes.	The car should be able to drive autonomously in the small scaled city in btw the two lines at all times.

Type: Feature	**Story Title:** Collision Avoidance System	**Priority:** 6
Product Backlog Item	**Conversation**	**Confirmation**
As an electronic specialist, I want the RC car to have a collision avoidance system so that it can guide the car when it encounters an obstacle.	Different types of sensors are going to be implemented to the microcontroller of the RC car so it can detect the obstacle in advance and respond in order to avoid the obstacle on the road	The autopilot of the car should avoid all obstacle confronted in mini scale city.

Type: Feature	**Story Title: Reflect Stop Signs Using Internal Map to RC Car, #7**	**Priority:** 7
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I want to program the software such as the RC car can distinguish Stop Signs so that the RC car can come to a full stop at different Stop Signs and obeys traffic rules.	The Stop Sign feature will be implemented using OpenGL into the Microcontroller. The Internal Map is a simulation of the real mini-scaled city which tracks the RC car in the OpenGL software. Once the smart care approaches a stop sign in the internal map, the software will communicate with the microcontroller which will decrease and stop motors for it to come to a complete stop in the none-simulated mini-scaled city.	The RC car should distinguish a Stop Sign and come to a complete stop.

Type: Feature	**Story Title:** Main Functionalities of Driving	**Priority:** 8
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I want the RC car to implement main functionalities of driving so that it can it drive itself properly around the city while obeying traffic and safety rules.	The microcontroller will need to send and receive data from our sensors and then feed it to the actuators to guide the car.	The RC car should be able to drive at constant speed, accelerate and reducing the speed properly while changing and turning lanes, and (optional) parking.

Type: Feature	**Story Title:** Internal GPS	**Priority:** 9
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I want the RC car to drive from point "a" to point "b" so that I can have internal GPS for the autopilot.	The RC car will need different type of sensors and programming so that can evaluate different route options in the mini-scaled city.	The RC car should be able to drive from the location entered to the location destined autonomously.

Type: Feature	**Story Title:** Safe Driving Standards	**Priority:** 10
Product Backlog Item	**Conversation**	**Confirmation**
As a programmer, I want the RC car to obey safe driving standards so that the autopilot simulates a mini-scaled real life autopilot.	I will need to implement an algorithm on the microcontroller to send and receive data from our sensors and then feed it to the actuators that will control the RC car safely.	The RC car will have a maximum safe limits for acceleration and steering.

Type: Feature	**Story Title:** Disturbances	**Priority:** 11
Product Backlog Item	**Conversation**	**Confirmation**
As electronic specialist, I want to make sure that the RC car is resistant to the on-road disturbances (bumps) so that the autopilot can continue on working smoothly.	The RC car will need to adjust to the sudden changes in direction, velocity, acceleration. The controller will need to be configured to adjust to the sudden on-road disturbances.	The RC car should be able to maneuver properly even after encountering a bump.

Type: Feature	**Story Title:** Override System	**Priority:** 12
Product Backlog Item	**Conversation**	**Confirmation**
As an electronic specialist/programmer, I want to make sure that I can manually override the RC car autopilot system so that I can control the car in case of emergency.	Need to program and configure the microcontroller so that the autopilot including the sensors can be turned off so that the RC car can be controlled manually in case of emergency.	I should be able to turn off the autopilot of the RC car and be able to override it and control the RC car manually.

Table 10.1: Product backlog

10.2 Sprints

Sprint 1:

<u>Objective: </u>Research on the hardware needed to build the RC car and the software associated to the autopilot.

Sprint 1 backlog

#Product Backlog Item (Priority Item)	**Task**
Autopilot, #1	Research on the implementation of the Autopilot ▪ Determine the most efficient way to implement the autopilot on the RC car ▪ Processing alternatives ▪ Sensing alternatives ▪ Research on the real-time operating system that works on raspberry pie, Autopilot code that works on raspberry pie (language, libraries) ▪ Research on the implementation of the Image processing ▪ Research on either if it should be performed on car or PC ▪ Determine the Bluetooth hardware and software compatible with raspberry pie

RC car & City Research, #2	Research on the hardware associated ▪ Research on different types of sensors ▪ Research on the hardware of the RC car ▪ Understand on how the RC car functions and make links between the hardware of the RC car associated with the software component of the autopilot. ▪ Design the mini scaled city ▪ Determine the appropriate scale ▪ Determine the width of lanes ▪ Determine the materials needed to design (# of foam pieces, material for lanes, obstacles) ▪ Finding the right the scale of the RC car

Product backlog items progress

Product Backlog Items Completed	**Modification and Results**
Autopilot, #1	▪ Determined the types of sensors that might be used(ultrasonic and gyroscope), how the image processing will be implemented via Bluetooth and the software associated ▪ Determined the type of processing that will be used for the autopilot (microcontroller -raspberry pi) ▪ Researched on the real-time operating system that works on raspberry pie, Autopilot code that works on raspberry pie (language, libraries)
RC car & City Research, #2	▪ Sensors determined ▪ Determined the RC car scale: 1/18 ▪ Determined the power requirement ▪ Did not finalize city plan ▪ Research on the hardware of the RC car completed in terms of how it functions(servos, motors, power & connections)

Table 10.2: Sprint 1 Schedule

Sprint 2:

Objective: Finalize city plan, design the schematic of the RC car, determine the power consumption of each components and design the simulation for the autopilot.

Sprint 2 backlog

#Product Backlog Item (Priority Item)	**Task**
RC car & City Research, #2	▪ Finalize the city plan
Building RC car & City, #3	▪ Design professional schematics ▪ Design table with components with budget that will used in the building of the RC car ▪ Power consumption of every components ▪ Ensure compatibility between components. ▪ Determine an RC car with rechargeable battery

Autopilot Simulation, #4	▪ Design an interface of an exact replica of the mini-scaled through with Open GL -City is designed in paint and by using Open GL, the mini-scaled city is simulated.

Product backlog items progress

Product Backlog Items Completed	**Modification and Results**
RC car & City Research, #2	▪ City has been designed
Building RC car & City, #3	▪ Schematic isn't finalized ▪ Designed power consumption table including tables and components, ▪ Found rechargeable RC car
Autopilot Simulation, #4	▪ Designed an interface of an exact replica of the mini-scaled through with Open GL

Product Backlog Items Grooming

Autopilot Simulation, #4	Added to the product backlog for sprint 2.

Table 10.3: Sprint 2 Schedule

Sprint 3:

Objective: Design a shield for the raspberry pi, add features to the autopilot simulation and identify lanes with image processing for the mini-scaled RC car.

Sprint 3 backlog

#Product Backlog Item (Priority Item)	**Task**
Building RC car & City, #3	▪ Finalize schematics and Design shield ▪ Ensure compatibility between components
Line Alignment, #5	Implement line alignment feature: ▪ Research on how to identify street lines through image processing so that the RC car can distinguish between street lanes. ▪ Create Pseudo-code, translate pseudo-code, Testing & Debugging.
Main Functionalities of Driving, #8	Implement feature: ▪ Research on how to Program the RC car in the Open GL that strictly obeys traffic rules such as stop signs. ▪ Create Pseudo-code, Translate pseudo-code, testing & debugging.
Internal GPS, #9	▪ Design internal GPS in the simulation of the RC car in the Open GL that follows a precise trajectory in the city as ordered.

Product backlog items progress

Product Backlog Items Completed	**Modification and Results**
Building RC car & City, #3	▪ Schematic has been designed and design shield remains unfinished.
Line Alignment, #5	▪ Street lines can be distinguished through image processing. However, it has not yet been implemented to the RC car.
Main Functionalities of Driving, #8	▪ Whenever there is a stop sign, the RC car in the Open GL comes to a complete stop. Not implemented in the RC car.
Internal GPS, #9	▪ Internal GPS is implemented in Open GL as the RC car can displace from any point a to any point b in simulated city.

Table 10.4: Sprint 3 Schedule

Sprint 4

Objective: Design shield for the raspberry pi, building the RC car and city. Lastly, start implementing features via the microcontroller to the RC car.

Sprint 4 backlog

#Product Backlog Item (Priority Item)	**Task**
Building RC car & City, #3	▪ Build the mini-scaled city ▪ Build the RC car ▪ Design the PCB for the RC car ▪ Connect the components together according to the schematic ▪ Test the RC and mini-scaled city compatibility.
Line Alignment, #5	▪ Implement Line Alignment using Image Processing into the Microcontroller
Collision Avoidance System, #6	▪ Implement Collision Avoidance System into the Microcontroller using three Ultrasonic Sensors ▪ Test the range and accuracy of the Ultrasonic sensors ▪ Determine the best location on the RC in order to detect objects around its circumference.
Reflect Stop Signs Using Internal Map to RC Car, #7	▪ Implement using OpenGL into the Microcontroller • Detect the Stop Signs ahead of time • Ensure the RC comes to a complete stop
Main Functionalities of Driving, #8	▪ Implement Main Functionalities of Driving feature into the Microcontroller. • Establish a constant speed using hardware PWM. • Maintaining proper deceleration and acceleration during turns and stop signs • Ensure the servo is turning the RC car smoothly using hardware PWM.
Internal GPS, #9	▪ Implement Internal GPS feature using OpenGL with the help of the speed encoder into the Microcontroller. • Hall sensor will detect the distance travelled which will help us track the RC car in the map. With the tracking available, the internal GPS will be implemented using OpenGL.
Safe Driving Standards, #10	▪ Implement Safe Driving Standards feature so that the RC car obeys traffic rules.
Override System, #11	▪ Implement Override feature into the Autopilot in case of emergency. ▪ Use up, down, left and right keys from the computer for emergency control • Establish a WIFI connection with the RC car through the Raspberry Pi.

Product backlog items progress

Product Backlog Items Completed	**Modification and Results**
Building RC car & City, #3	▪ The RC car was successfully designed according to our preference and is compatible with the mini-scaled city in terms of driving.
Line Alignment, #5	▪ Street lines are distinguished through image processing and the autopilot can maneuver in between lanes.
Collision Avoidance System, #6	▪ The RC car has the capability to determine when an object is located in its frontal side and come to a complete stop.
Reflect Stop Signs Using Internal Map to RC Car, #7	▪ The stop signs in the mini-scaled city stops the RC car whenever it the traffic sign is nearby.
Main Functionalities of Driving, #8	▪ The RC car maneuvers at constant speed and turns smoothly during turns due to the servo and motor being controlled with hardware PWM.
Internal GPS, #9	▪ Internal GPS is implemented and the smart can be maneuver from a location to any destination destined.
Safe Driving Standards, #10	▪ Traffic rules are followed strictly in the mini-scaled.
Override System, #11	▪ The smart can be controlled using up, down, left and right keys from the computer for emergency control when the autopilot malfunctions.

Table 10.5: Sprint 4 Schedule

10.3 Updated Tasks Breakdown

Task1: Research on Microcontrollers, Sensors, Auto-Pilot and Image Processing

Sub-Task1: Bridging the knowledge gap with the help of research, Brainstorming

Sub-Task2: Comparing different Microcontrollers based on decision matrix and then choosing the optimal one

Sub-Task3: Comparing different sensors and Image processing techniques based on decision matrix and then choosing the optimum one

Sub-Task4: Autopilot analysis for Collision Avoidance and to guide the car from one point to another

Sub-Task5: Budget Estimation Sub-Task6: Project Planning

Task2: Assembling the RC Car and building the miniature city model

Sub-Task1: Calculations for power consumption for microcontroller and other components Sub-Task2: Designing schematic, mini-scaled city and as well as the shield.

Sub-Task3: Connecting sensors, actuators to the microcontroller

Sub-Task4: To include different test cases in the mini city model that would be used by the smart to perform different functionalities accordingly.

Deviation: Needed to design schematic for the RC car, a mini-scaled city for the RC car to drive upon and as well of designing a PCB. Removed the Shield for the motor and replaced it with an Electronic Speed Controller (ESC).

Task3: Implementation

Sub-Task1: Using image processing software to recognize different subjects

Sub-Task2: Implement the Internal map of our miniature city in autopilot

Sub-Task3: Programming the autopilot on microcontroller for collision avoidance and main functionalities of driving features.

Sub-Task4: Establishing a connection between computer and RC car using wireless connection.

Deviation: The interface of the autopilot is first designed in the Open GL. The internal GPS feature is first implemented in the map before the collision avoidance feature. This was just as a preference due to the fact that the RC car has not been built due to parts not being ordered yet. Also, added main functionalities of driving as a feature with collision avoidance.

Task4: Testing/Debugging

Sub-Task1: To debug the sensors to know if they are providing accurate data Sub-Task2: To test the autopilot for different cases in our city model

Sub-Task3: Ensuring successful communication between sensors, microcontroller and actuator Sub-Task4: Ensuring connection between computer and the car

Deviation: The total weight for the Smart Car was too heavy and the RC car was no table to drive itself in the mini-scaled city as the motor load torque was not sufficient and the servo had a mechanical and hardware problem by default. Hence, the team had changed to a bigger RC car which prompted us to change the city in order for it to be compatible in terms of size. However, the new RC car had a failing servo and motor due to being seven years old (belonged to member in team 18). Moreover, the peak current of the motor was pulling too much current through the motor driver which prompted to change the entire design of the hardware which lead us to the Electronic Speed Controller (ESC).

10.4 Tasks Allocation

Task	Sub-Task	Person Responsible
1	1	Simaar, Jafar, Talha, Muneeb
1	2	Jafar
1	3	Muneeb, Talha
1	4	Jafar
1	5	Simaar, Jafar, Talha, Muneeb
1	6	Simaar
2	1	Simaar,Muneeb
2	2	Simaar, Muneeb
2	3	Muneeb, Simaar
2	4	Simaar
3	1	Talha
3	2	Jafar
3	3	Jafar. Talha, Muneeb
3	4	Talha, Jafar
4	1	Muneeb
4	2	Talha, Jafar
4	3,4	Simaar, Talha, Muneeb ,Jafar

Table 10.6: Tasks allocation

10.5 Deviations

1) Autopilot simulation interface added to the product backlog item.
2) Research for the city and building the city have been to the product backlog item.
3) Features from the product backlog items will take more time
4) Image processing is much complex than expected and requires more time.
5) Schematic needed many corrections due to component compatibilities issues.
6) Designing the shield has been removed from the task.
7) The PCB had to be revised.
8) Assembling the RC car was problematic. First smart car had mechanical and hardware problems with the servo and was not able to push the total assembled weight of all the hardware. Second smart car has mechanical problems with the servo and a failing motor and servo due to being an old RC car belonging to a member of the team. Also the older RC car was bigger in scale in order to push the total weight. However, as being a bigger RC car, the motors possessed a higher peak current which was too high for our system which prompted us to the change the entire design of the hardware.

10.6 Scheduling

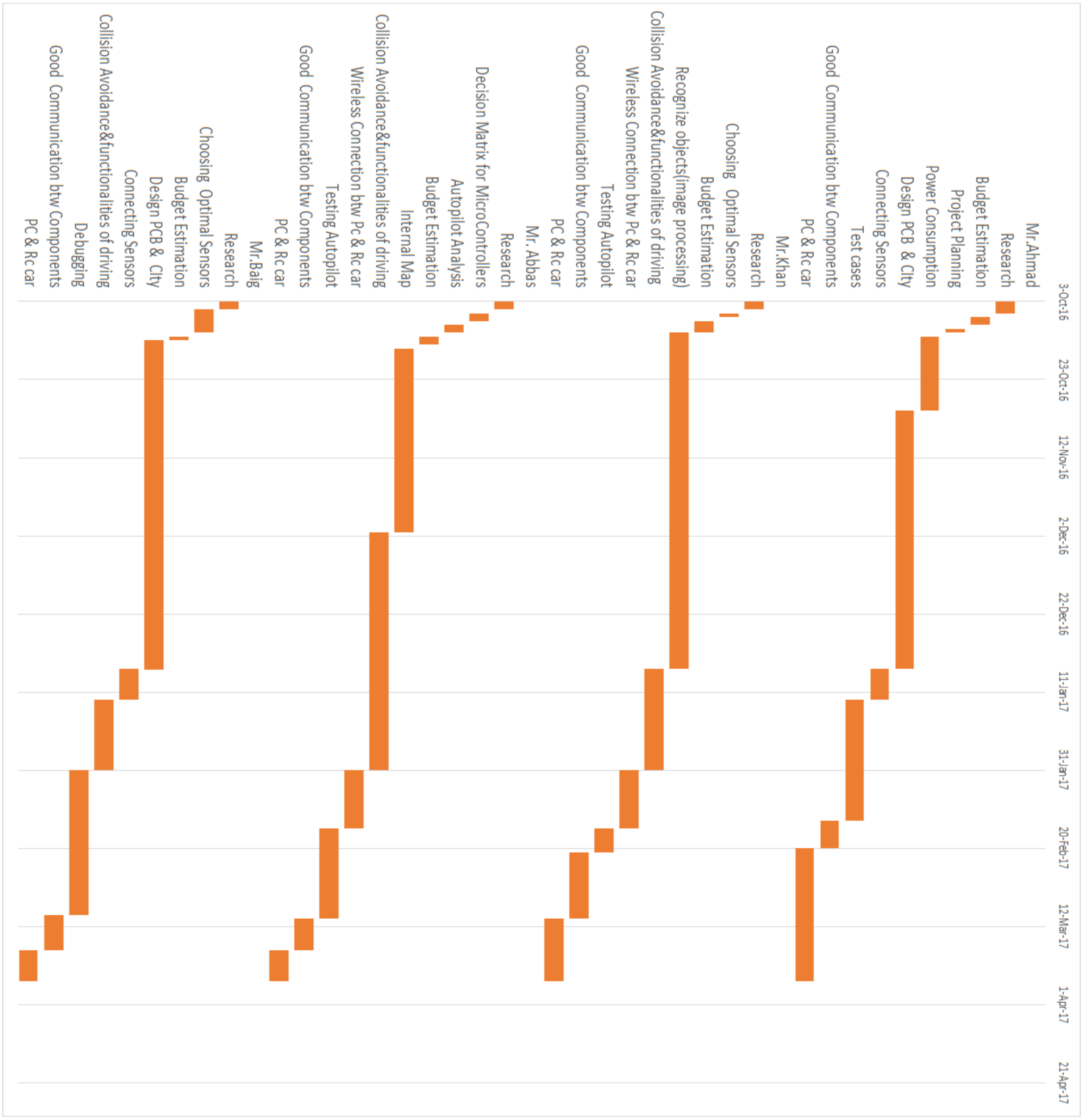


Figure 10.1: Updated Scheduling

10.7 Budget

Component	Amount	Cost ($)
Raspberry Pi 3B	1	69.99
Pny Elite 16GB microSD Card with Adapter	1	10.34
SINOELE Mini Power Bank	1	29.99
RC Car	1	180.00
2S Lithium Polymer Battery(LiPo)	1	84.99
LiPo Charger	1	34.99
HC-SR04 Ultrasonic sensor	3	16.99

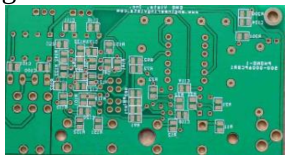
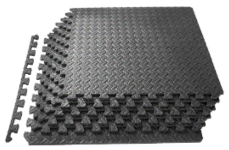
Hall sensor	1	7.99
SainSmart Camera Module	1	23.99
Neodyme Magnets Super 0.3`10p	3	12.64
Through-Hole Printed Circuit Board 	1	0 (Provided By School)
ProSource Puzzle Exercise Mat EVA Foam Interlocking Tiles 	3	30.32
Raspberry Pi 3 Model B case, KuGi	1	8.5
So-Myshtech 170 Points Mini BreadBoard for Arduino Proto Shield	1	11.99
Haobase 120 pcs Multicolored 40 pin male to Femal & 40 Pin Male to Male	1	10.99
Total cost without taxes		**CAD 594.40**
Total cost with taxes		**CAD 683.56**

Table 10.7: Total Budget

Conclusion

Inspired by Google, Uber and Tesla, car manufacturers are all working on their own versions of an autonomous vehicle which are economically and socially beneficial to both manufacturers and their customers. Due to the safety-critical environment in which vehicles operate, the complexity of the technology and the numerous ethical, regulations and law-abiding challenges, the tasks of building a driverless car are immense. The aim of our project is to manufacture a smart car that allows safe autonomous travel from one point to another.

The smart car operates through an Open GL simulator and a Raspberry Pi. The RC car uses Image Processing to detect the lanes, Ultrasonic sensors to detect obstacles and a Hall sensor to detect the speed the car is travelling through our mini-scaled city.

In the final stage of our project, we were able to produce a smart car that has the capability to drive from any location entered from the user to the desired locations autonomously while respecting and abiding the law and traffic rules.

REFRENCES // in progress

[1] N. Administration, "FARS Encyclopedia", Www-fars.nhtsa.dot.gov, 2016. [Online]. Available: http://www-fars.nhtsa.dot.gov. [Accessed: 30 - Dec- 2016].

APPENDICES

APPENDIX A: CEAB Graduate Attributes Part 1

Analogical Case Study:

Self-Driving cars use to be a futuristic idea few years ago. Today, the self-driving is in the development phase. Uber, a company involved in car transportation and food delivery, has recently change its corporate trajectory with highly sophisticated self-driving vehicles. In order to increase the margin of profits, Uber has decided to invest profoundly in the autopilot technology so it could one day make its vehicles autonomous and get rid of their employees also known as the drivers of Uber.

Due to the autopilot technology still being in the development phase which has created many problems for Uber. As an example, during the week of March 6, the self-driving Uber drives were able to drive more than 0.8 miles autonomously before the driver of the car had to take over the autopilot for safety reason [1]. This clearly demonstrate the current state of the technology still being in the beta. As a result, even though the public has a positive sentiment towards Uber, they are still not ready to embark in this incredible journey as there are still safety concerns. Another great concern was regarding an event where an Uber autonomous car ran 6 red lights in California [2]. This is another example that is greatly concerning towards public safety.

Currently, there are still no laws regulating the autopilot and many governments or states in the United-States are currently silent [3]. The main questions that arise from the autopilot are safety side in terms of the passengers. For now, as it is in the development phase and due to the recent problems with Uber, it could be seen as unsafe. However, when we do the math and compare the autopilot vs man driven cars, it is only a little superior to the man driven cars according to the "Green Car Reports" [4]. Moreover, there are many questions that arise in terms of legality as who is liable if the autopilot strikes a pedestrian? [5] Whether a child can use it and what should be the legal age to be the main driver ("passenger") in the autopilot. For now, there are no straight answers and might get an answer after lawsuits that might arise due to these problems.

In our capstone project, the autopilot is much smaller scale of a RC car. We cannot directly really address these dilemmas and cases in our project as it cannot be applicable due to simplicity of our autopilot. The smart car project has very limited capabilities wise when compared to Uber`s self-driving cars. However, we are planning to implement an autopilot RC car which does in fact obeys the Canadian laws such that our self-driving car will be following strict regulations and obeying traffic laws in our mini- scaled city. Uber is not environmentally friendly and our aim is to make our smart car powered through a sustainable resource. We also cannot apply any other social or legal links between our project autopilot.

Contemporary Practice:

As Uber and Lyft have become an international sensation in the taxi/ride sharing services, Tesla whom manufactures the Autopilot system in their vehicles, is planning to launch "Master Plan Part Deux" which is a taxi/riding service very similar to both Uber and

Lyft. However, unlike Uber and Lyft, the "Master Plan Part Deux" is entirely autonomous which is completely different from any other taxi services currently offered.[6] The problem right now is that there are a lot of legal, social and moral issues regarding the Autopilot itself even though it is legally accepted as explained in the Analogical Case Study and this technology is still in development.

Hence, before even launching the "Master Plan Part Deux", these issues will need to be addressed before replacing any Uber or Lyft as customers will be driven towards it.

Moreover, Autopilot in the recent days have gained a lot of criticism due to malfunctions that have happened on the roads and has caused deaths in such cases with Tesla. Unlike Uber which possesses a slightly advanced autopilot system over Tesla, Teslas vehicles are powered through a lithium battery and is powered through electricity rather than gasoline which makes Tesla a sustainable and environmental friendly company. Another example of Tesla being sustainability company economically and environmentally is that its currently creating a Gigafactory in a dessert in Nevada, United-States which will produces lithium-ion batteries through solely based on solar radiated energy.[7]

Originally we want to work on a capstone project that had something to do with autopilot. Either could had been a drone or boat. Due to the autonomous vehicle being a topic covered numerous times in the news outlets, we ended up decided to work on an autonomous vehicle, being the first group to do so in Electrical and Computer engineering department at Concordia University. We ended up looking upon both Tesla, Uber and even Google due to their highly sophisticated autopilot hardware and software. However, we decided to take few ideas such from both Tesla and Uber and competing our puzzle which is our capstone. We took the notion of sustainability from Tesla and decided to apply it to our own project making our car smart but also a "green car". We liked the notion of self-driving taxis from Uber and hence also took this idea and applied it to our project as it will be able to move from any locations in our mini-scaled city to any destination set by us.

The complexity in self-driving vehicles are not just found in its software and hardware design but in many other forms such as if it respects and remains in the boundaries of the law and regulations set by the government, such that it conducts in a proper ethical manner and such that it has potential of being both economically and environmentally friendly. Our project is just a mere simulation. Therefore, it will not be directly affected by such complexity.

REFRENCES FOR CEAB GRADUATE ATTRIBUTES PART 1

1- "Uber's autonomous cars drove 20,354 miles and had to be taken over at every mile, according to documents", Recode, 2017. [Online]. Available: http://www.recode.net/2017/3/16/14938116/uber-travis-kalanick-self-driving-internal-metrics-slow-progress. [Accessed: 22- Mar- 2017].

2- M. Geuss, "Uber's self-driving cars ran through 6 stoplights in California, NY Times says", Ars Technica, 2017. [Online]. Available: https://arstechnica.com/cars/2017/02/ubers-self-driving-cars-ran-through-6-stoplights-in-california-ny-times-says/. [Accessed: 22- Mar- 2017].

3- *Hg.org*. [Online]. Available: https://www.hg.org/article.asp?id=31687. [Accessed: 11- Jan- 2017].

4- D. Noland, "How safe is Tesla Autopilot? Parsing the statistics (as suggested by Elon Musk)", *Green Car Reports*. [Online]. Available: http://www.greencarreports.com/news/1106613_how-safe-is-tesla-autopilot-parsing-the- statistics-as-suggested-by-elon-musk. [Accessed: 11- Jan- 2017].

5- [3] "Autonomous Cars: The Legality of Cars on Autopilot", *MTTLR*. [Online]. Available: http://mttlr.org/2015/04/08/autonomous-cars-the-legality-of-cars-on-autopilot. [Accessed: 11- Jan- 2017].

6- J. Dow, "Tesla's new self-driving car can only make you money on the ride-sharing 'Tesla Network', not Uber or Lyft", *Electrek*. [Online]. Available: https://electrek.co/2016/10/19/teslas-new-self-driving-car-can-only-make-you-money-on-the-ride-sharing-tesla-network-not-uber-or-lyft/. [Accessed: 11- Jan- 2017].

7- "Gigafactory 1", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Gigafactory_1. [Accessed: 22- Mar- 2017].

APPENDIX B: CEAB Graduate Attributes Part 2

PUBLIC PERCEPTIONS OF TECHNOLOGY

Content analysis of major media sources for public information about the innovation

The autopilot RC car that the team worked for the capstone is a mini-scaled simulation of the autonomous vehicles which is both seen as prosperous and innovative, and dangerous and unsafe.

Many major media sources publicize the idea that the autonomous vehicles are not fully reliable and safe due to the small percentage of the Tesla smart cars that have resulted in car accidents because of the autopilot malfunctioning. These major media sources convey their message through illustrations of the accidents. The DailyMail as is a great example of major media source that uses pictures of the accidents, footage of the failure of the software and emotional manipulation to demonstrate the ineffectiveness of the technology behind the autonomous vehicle.[1] However, many other major media sources are pushing a positive image of the autopilot system through safety. The Guardian as an example view the self-driving cars as a game changer in which the world would become safer world with just autonomous cars and how it's bringing a revolution. [2] Hence, the major media sources are both pushing a positive and negative view on self-driving cars through different methods such as illustrations or safety statistics.

Research to assess public concerns about, and aspirations for, the development and application of the innovation.

The autopilot system is a very hot topic in the media, but there is still a lot of controversy regarding its safety and usage amongst the public.

According to Researchers Azim Shariff, Jean-François Bonnefon and Iyad Rahwan, who have been conducting research amongst the approval rate of the autonomous vehicles amongst the public, they concluded after analyzing the results obtained that the public even though they are in agreements that the autopilot system can save more lives on the road once applied, do not wish to obtain one in the future [3]. A brief example taken from the previous experiment was a question asked on whether the autonomous car should save the life of the driver or the many pedestrian the vehicles endangers if the vehicle is in such situation. Hence, the result of this scenario was simply save the greater good than saving the driver. [3]. Therefore, this research greatly demonstrates, even though the autopilot system has been greatly appreciated amongst the public, the public still have many concerns regarding self-driving cars on safety matters. Just as there is concern, there is also a great amount of the population that think fully positively of self-driving cars. According to Sahar Danesh of the Institute of Engineering and Technology, she believes that the autopilot cars are much safer than the traditional way of driving.[4]. There is a lot of statistic right now that demonstrating the efficiency and safety of the self-driving over the conventional way of driving. However, due to the autonomous vehicles still being in the development phase, there is still a lot of controversy amongst the public simply based on the safety and ethical sides of the system.

Survey research to identify public reaction to media portrayals of the innovation and to track changes in public attitudes about developments in the innovation.

A short survey was conducted in the University of Concordia regarding their views and regarding the autonomous vehicles. Two questions were asked during the survey, the first being: "What is your view regarding autonomous cars?" and the last one being :"Would you ever think of purchasing a self-driving car near the future ?". The results clearly showed a different sentiment towards smart cards against the analysis reported by major media sources. 71% of the participants, viewed the autonomous vehicle positively. However, 77% amongst the participants do not feel safe with the concept of smart cars and rather keep driving their own.

Map ethical, legal, social, and professional concerns for your particular project in media.

The introduction of the autonomous vehicle carries many positive prospects but as well as few concerns regarding its efficacy. The technology behind self-driving is still in the development phase which creates safety concerns. However, it is not dangerous as harshly and negatively portrayed by major media sources. It will one day become much more effective and safer than conventional way of driving. Moreover, various ethical dilemmas arises from this technology regarding life. One of the main question that arises from the technology behind self-driving cars is that if it should rather prioritize the life of the driver over the life of the pedestrians the car endangers in such situations. Due to this ethical dilemma, this created great concern and confusion amongst the public. Furthermore, there are currently no laws regulating the autonomous cars as its still in the beta version. Socially, it has the capability of saving lives and making the driving experience much more safer. In 2012, there were 2546 crash accidents and 58.8 % amongst these car accidents were due to the involvement of drugs and alcohol [5]. The autonomous vehicle could in general decrease these death tolls of its potential of becoming safer than the conventional way of driving once it`s out of its development phase and also mainly because the autopilot will not allow drivers under the influence to steer the car manually but rather the technology will take full control of the vehicle. Hence, the autonomous vehicles has the capability to decrease the number of death tolls and permit drivers under the influence to still be able to get to their destination safely.

TECHNOLOGY ASSESSMENT AND CHOICE

Identify choices and how choices have been influenced by the knowledge acquired through the application of the techniques described in the Real-Time Technology Assessment methodology.

Real-Time Technology Assessment methodology is great tool used to design projects by engineers by understanding the fundamental social, moral, economic and environmental implications. The main aim of our capstone project was to create an innovation that was environmentally friendly, that respected the country`s law, moral and ethical values and possessed a positive attribute towards social norms. While respecting the previous mentioned conditions, the autopilot technology was chosen as our capstone project which positively influences the economy and the community socially as it would car accidents and reduce traffic which is bad for the economy as fuel is wasted as the car stalls and time. As an example, in 2013 a study was conducted regarding the relationship between congestion and the economy by The Centre for Economics and Business Research, a London-based consultancy, and INRIX, a traffic-data firm, they observed four countries such as Uk,France, Germany and the United-Stated and reported a total loss $ 200 billion lost due to congestion alone.[6] One of our choices that we wanted to implement in our project was the smart to distinguish between different objects in our mini-scaled city such that it classifies objects according to it type (example: pedestrians vs obstacles) such that our project respects ethical and moral implications.

Reflect on and demonstrate explicitly iterations in the process followed where ethical, legal, environmental, economic and/or social implications were identified and addressed.

Self-conducting vehicles provides the safest standard of driving. As the autonomous vehicles have been portrayed as unsafe from major media sources, in order to change that view, in our mini-scaled , the smart car obeys traffic rules and regulations such that it respect legal implications.

Today, the majority of cars have an internal combustion engine which is a great emitter of CO_2, a greenhouse gas. Our vehicles runs on an electric motor which does not emit any greenhouse gases similar to Tesla, and we made great efforts to use to the least amount of components to make our project. Lastly, we wanted add a functionality to the smart car that could had differentiated between a person and an obstacle in our city so that it coordinates in the city such that it guarantees the safety to both the passenger in the autonomous vehicle and the pedestrians. Unfortunately this could had not been achieved due to certain constraints.

Evaluate the role of real-time technology assessment in their own (and other similar) projects. What choices were affected by this research? (

We made the requirements for the project by using the analogical case studies. Due to Tesla autonomous vehicles failing numerous times and getting into legal, moral and social dilemmas, it prompted us to make our project by avoiding such dilemmas. However, we also used their innovation to guide us in a positive way such as being environmental friendly. Also, our smart car uses Lipo as our source of power rather than normal 1.5 V batteries as it is more environmentally friendly and economically cheaper.

Tesla is a great example for innovation and sustainability. However, their autopilot system is not fully developed and has failed numerous times even though is still amongst the most advanced in its field . On the other hand, Uber`s autopilot is slightly better than Tesla`s.[7].

Since December 2016, Uber self-driving cars have driven more than 20000 miles in a week and has plans to one day increase their margin of profits by replacing their man conducted cars by simply a robot. Similar to Uber, we are able to set a destination in our mini-scaled city and our smart car self-drives to that location. However, due to Uber`s self-driving technology still being in the development phase and our project being a mere simulation, there is currently no way right now to bypass regulations and laws without a solid technological advancement in this field.

REFRENCES FOR CEAB GRADUATE ATTRIBUTES PART 2

1- "Footage shows what happens when Tesla autonomous driving goes wrong", *Mail Online*, 2017. [Online]. Available: http://www.dailymail.co.uk/sciencetech/article-3281562/Tesla-autopilot-fail-videos-emerge-Terrifying-footage-shows-happens-autonomous-driving-goes-wrong.html. [Accessed: 20- Mar- 2017].

2- S. Levin and M. Harris, "The road ahead: self-driving cars on the brink of a revolution in California", *the Guardian*, 2017. [Online]. Available: https://www.theguardian.com/technology/2017/mar/17/self-driving-cars-california-regulation-google-uber-tesla. [Accessed: 20- Mar- 2017].

3- [Online]. Available: 1- http://www.cbc.ca/news/technology/driverless-vehicles-ethics-1.3648029. [Accessed: 20- Mar- 2017].

4- D. Millward, "Tesla Autopilot crash: what next for driverless cars?", *The Telegraph*, 2017. [Online]. Available: http://www.telegraph.co.uk/cars/comment/tesla-autopilot-crash-what-next-for-driverless-cars/. [Accessed: 20- Mar- 2017].

5- "Statistics", *MADD Canada*, 2017. [Online]. Available: http://madd.ca/pages/impaired-driving/overview/statistics/. [Accessed: 20- Mar- 2017].

6- Economist.com, 2017. [Online]. Available: http://www.economist.com/blogs/economist-explains/2014/11/economist-explains-1. [Accessed: 21- Mar- 2017].

7- U. why, "Uber is wrong to compare its self-driving cars to Tesla's Autopilot — here's why", *Business Insider*, 2017. [Online]. Available: http://www.businessinsider.com/why-uber-shouldnt-compare-self-driving-cars-to-tesla-autopilot-2016-12. [Accessed: 22- Mar- 2017].

8- "Uber’s autonomous cars drove 20,354 miles and had to be taken over at every mile, according to documents", Recode, 2017. [Online]. Available: http://www.recode.net/2017/3/16/14938116/uber-travis-kalanick-self-driving-internal-metrics-slow-progress. [Accessed: 22- Mar- 2017].