جامعة الأمير محمد بن فهد

PRINCE MOHAMMAD BIN FAHD UNIVERSITY

College of Engineering

Department of Electrical Engineering

Spring 2018-19

Senior Design Project Report

Smart Electrical Car

In partial fulfillment of the requirements for the Degree of Bachelor of Science in Electrical Engineering

Team Members

	Student Name	Student ID
1	Nawaf Almansour	201301171
2	Hassan Alghamdi	201301928
3	Mahdi Alfarwan	201002050
4	Atif Alqahtani	201402544
5	Abdulhadi Alhajri	201500089

Project Advisors:

Advisor Name: **Dr. Chedly B. Yahya**

Abstract

The world today suffers from a mass issue due to increased gas emissions that is seen due to carbon emissions from cars. Moreover, people with disabilities necessitate the addition of "smart electric cars" to the working market. Due to the magnitude of the problem that gas emissions present, leading companies like Mercedes with "The Smart Car" idea have already released patents. The cars can cater to motorist desires along with environmental benefits. However, with the addition of smart electric cars, the problem of gas emissions persists, as the car cannot travel to distant places with a single charge. Moreover, the electric source around the world and in Saudi Arabia still relies on methods that lead to increased carbon emissions. As a result, it is necessary that the electric power generation methods are green in effect. People with disabilities will be able to benefit from the car, as the law will permit safe travel with their condition. A view also studies the effects of electric cars on the environment, electricity consumption, average mileage, and the emissions over a fixed period. Studies show that conventional car create two times carbon emissions as compared to electric cars. With a 10-year lifecycle, the traditional car will emit 66,000 pounds more carbon than the smart electric car (Van Vliet et al., 2011). Moreover, with the addition of a solar panel, electric cars can power electric cars more efficiently. The smart car can also curb traffic-related accidents due to drivers' negligence or in a state of unconsciousness due to lack of sleep or intoxication. Today, there are many convertible and other options that allow the user to undergo the full experience. Due to the magnitude of problem that carbon emissions and driver negligence proposes with traditional vehicles, smart electric cars are a highly beneficial alternative.

Table of *Contents*

1. Introduction

1.1 Project Definition

To design a smart electric car that will be able to

a) follow a lane in a real road with straight and curved parts,
b) detect obstacles and maneuver around them,
c) recognizes and follows traffic lights

1.2 Project Objectives

1) Simulate the latest advanced technology of the actual self-driving cars
2) Reduce the need for the human control interference
3) Expand the local content of smart systems inventions to be competitive with the global race
4) Increase the duration cycle of the preventive maintenance
5) Minimize gas emission

1.3 Project Specifications

1) Runs on rechargeable battery for one hour
2) Size approximately: length (50 to 100 cm), width (30 to 50 cm) and height (40 to 60 cm)
3) Detects lane borders, turns and drives near lane center
4) Recognizes obstacles and maneuvers around them
5) Recognizes and follows traffic lights

1.4 Product Architecture and Components

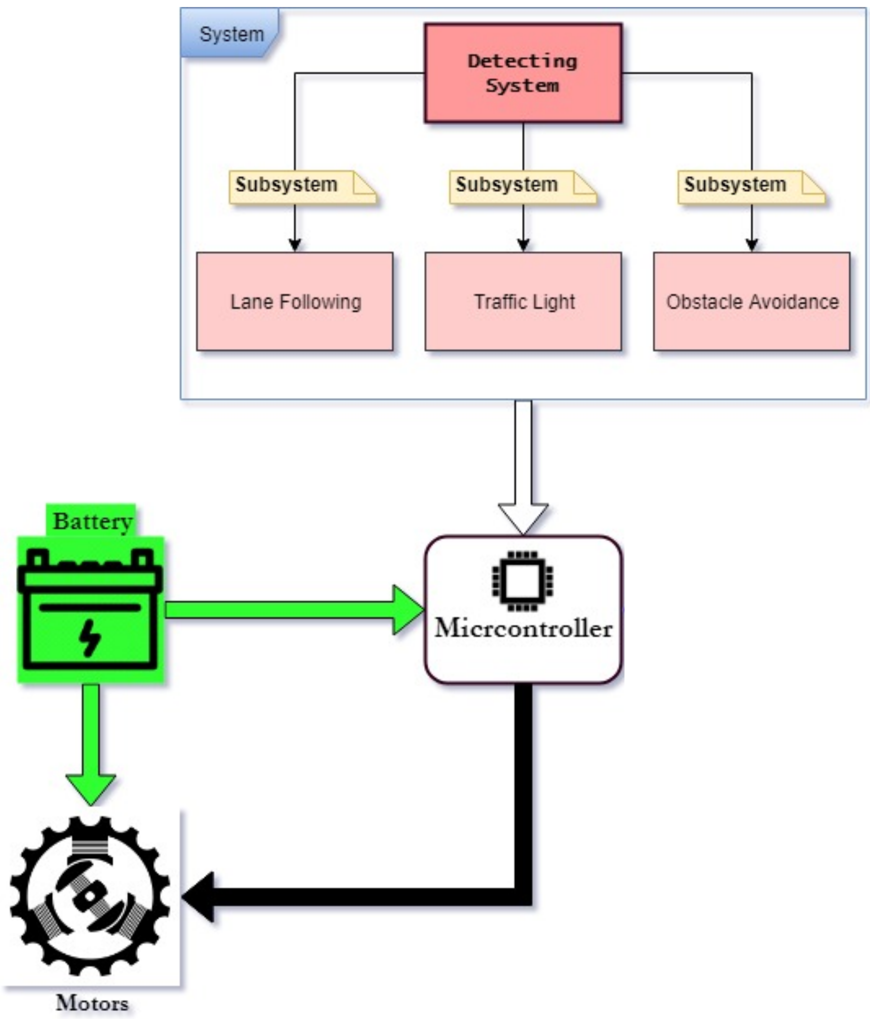


Figure 1 Smart electric car integrated systems block diagram

Microcontroller: in our project we are using two microcontrollers. Raspberry Pi and Arduino

Detecting system: under the detecting system we have four sub-systems

1. Lane following.
2. Traffic light detector.
3. Obstacles detector.

1.5 Applications

The proposed project can be used:

- In city roads for transport people from place to another.
- In industrial to deliver specific thing.
- By handicapped who have disability prevent them from driving

2. Literature Review

2.1 Project background

With over 110 million vehicle related accidents in the U.S. since 2000, engineers are yet to make a breakthrough with smart cars. Particularly, there is a requirement to dominate safety mechanisms, automatic cruise control buttons, laser systems and ultimately preventing collisions. Engineers face a huge gap during stabilizing control and assisting by automatically applying breaks. Engineers are yet to make progress in discovering “intelligence transportation” (Helmers & Marx, 2012). Overall, smart cars operate on a laser based sensor that looks at slowing down when another vehicle approaches. Progress is required for intelligent systems to be accepted as mainstream riding options. Researchers suggest that creating and maintaining maps for smart cars is critical work and requires clarification based on autonomous technology.

In order to overcome key issues, it is imperative that engineers and robotics are used to drive the car through many complex interactions that are difficult for robots today. Cars require training, and interpretation that pedestrians and other cars demonstrate on the road are being used. Moreover, to mitigate the effects that are seen with weather changes, cars require further engineering changes and sensor building. Some issues that are present with self-driving cars relate to regulatory bodies in the country as well, and they can be evaded to work for a unanimous support of electric cars. One can also relate key issues present in cybersecurity that is linked to smart electric cars (Lassila, Haakana, Tikka & Partanen, 2012). However, the best solutions relate to improving cybersecurity issues by ensuring that these cars are impenetrable.

Previously, no key solutions have been offered to people with disabilities or in cases when the drive is intoxicated (Musselwhite & Haddad 2010). As a result, millions of lives have been lost. With the help of smart electric cars, one can ensure that loves are saved, an automatic transition is made, and that drivers are presented in a safe driving format.

2.2 Previous Work

1) Self-Driving Car, Prince Mohammed Bin Fahd University, Spring Semester 2017-2018

Figure 2 PMU Self driving car

- Design self-driving car that can drive between two predetermined locations by following a line track.
- Design a manual control mode via a smart phone application in emergency.

2) Self-driving car TOY model, University of Vermont July, 2015 (Bogdan Djukic)

As shown in figure 3.

Figure 3 Bogdan Djukic car

- Designing a robotic operating system running with obstacle avoidance system
- Design to run between multiple locations

2.3 Comparative Study

Table 1 shows the features of each of the projects mentioned before. In addition, it shows our project in order to compare between them and show the differences and similarities.

Table 1: comparing our project with the previous projects

Projects	1	2	Our project
Obstacles avoidance	√	√	√
Drives near lane center	X	√	√
Recognizes traffic lights	X	X	√
Recognizes stop signs	X	X	√

As we can see in table 1, our project has the ability to detect an obstacle and take the right decision by stopping until the road become clear and safe again to continue driving. Another feature that can be found in our project is the driving near lane center, which makes it look like a real car on the road. The feature that makes our project special is the ability to recognize the traffic lights and stop signs and make the right decision according to what it detect. Project 2 which made by Bogdan Djukic have to similarities with our project, obstacles avoidance and the lane follower. Moving to project 1, the only common system between it and our project is the obstacle avoidance.

3. System Design

3.1 Design Constraints

Safety:

The car should operate safely with acceptable speed without losing the track or hitting objects.

Environmental:

Electric cars have no fuel to burn, therefore no emission of CO 2 and it works on clean environment.

Social:

The system will help people to save the driving time which allows them to be more productive while the car drive itself.

3.1.1 Design Constraints: Safety

Each year 30,000 to 40,000 people killed on the roads in the KSA. Despite a recent decline, there were 247 thousand road accidents in the KSA in 2016 (latest data from the KSA traffic police department/ ministry of interior). Most of these accidents are caused by driver' errors. Driver's mistakes are due to lack of knowledge, failure to comply with the traffic regulations, road distractions, or health issues while driving. All the parts that have been used are considered to be safe to deal with which will be operating with low voltage DC power supply and properly insulated from being exposed which might cause electric shocks or short circuiting.

3.1.2 Design Constraints: Environmental

Since the system will reduce the need of human interference, which allows robotics to make smarter decisions which reduce the travelling distance, accidents, and the number of cars needed for individual's transportation. Eventually, this will reduce the CO_2 emissions produced by increased travelling distances and the number of cars for each individual. Also, since the number of accidents will decrease, this will enhance the environmental losses due to loss of manufacturing resources.

3.1.3 Design Constraints: Social

The system will help people to save the driving time, which allows them to be productive while the car drives. Some studies have been conducted on the influence of robotics over the local society of KSA, which concluded that the society would be able to accept the robotics driven technologies. Yet, there are still some social constrains including convincing the local authorities about the visibility and safety of the self-driven cars. Therefore, we recommend doing further future improvements in developed countries since they are more flexible and adaptive about new technologies.

3.2 Design Methodology

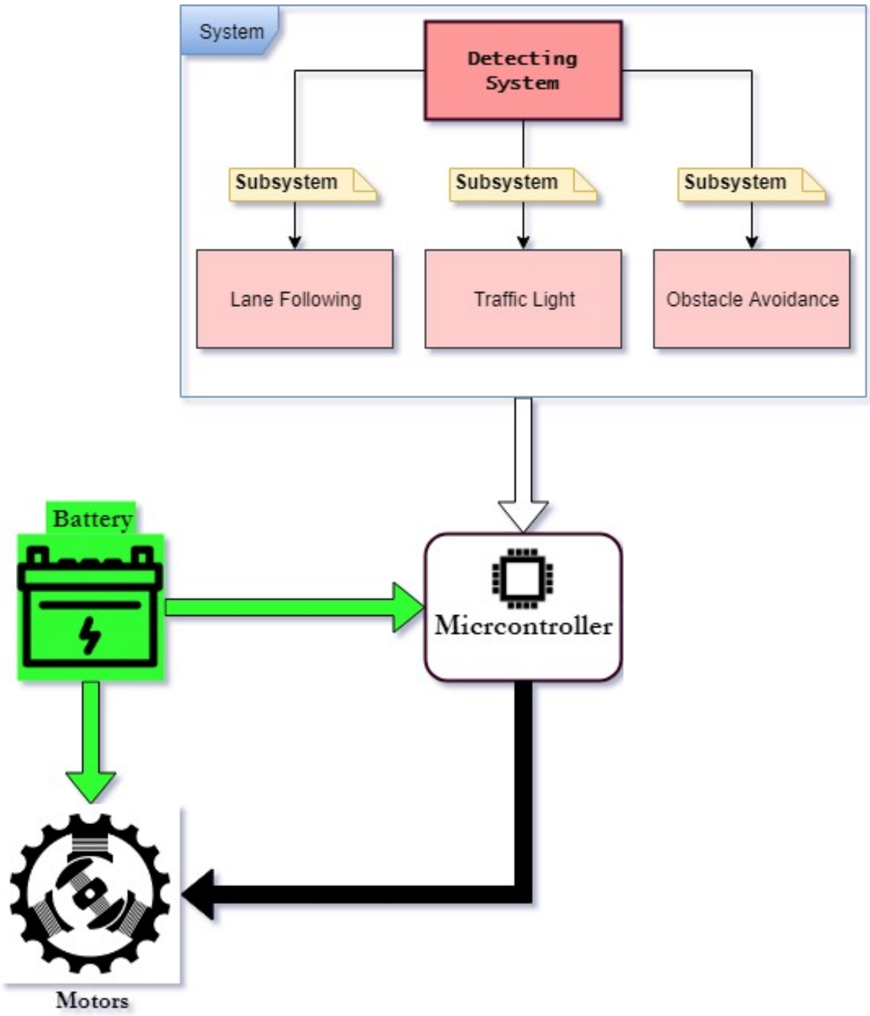


Figure 4 Smart electric car integrated systems block diagram

Phase 1 (Fall 2018-19)

- Search about the project
- Identifying project subsystems
- Verify the components are available in PMU or local market
- Write a project management plan with a list of tasks distributed
- Acquire all needed components
- Test the components and make sure they work
- Start building subsystem 1 (Lane Following)

- Test Lane Following

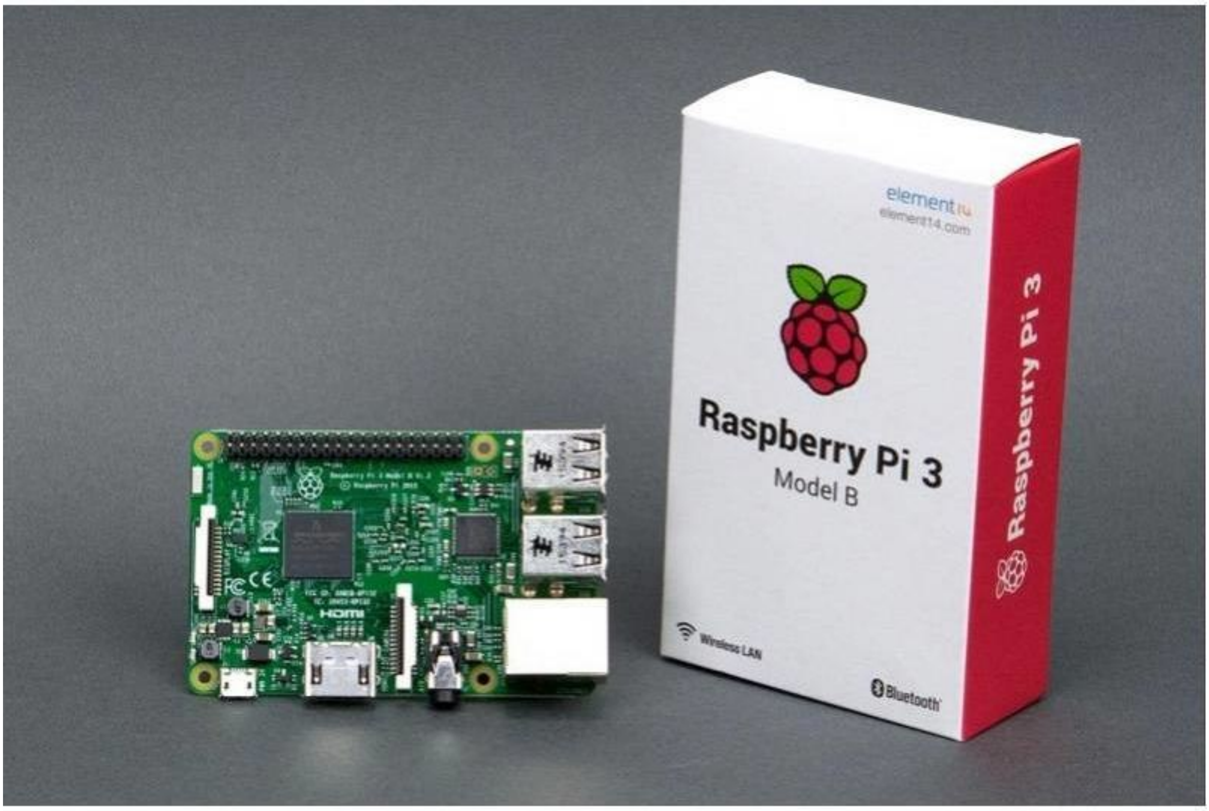
Phase 2 (Assessment III)

- Start implementing subsystem 2
- Test Subsystem 2
- Integrate subsystem. 1 with subsystem 2
- Start implementing subsystem 3
- Test Subsystem 3
- Integrate all subsystem

3.3 Product Subsystems and Components

In this section will design the subsystems and select the components according to our desired specifications.

3.3.1 Product Subsystem1: Lane Following



Raspberry Pi will be used to image processing and transfer it to the pc, the camera will be connected to the raspberry pi and it will transfer the image to the pc. The Raspberry Pi is the core component in our project, it is used in the traffic light, stop sign and lane follower subsystem.

We compared the raspberry pi to the Arduino mega as shown in the following table

Raspberry Pi	Arduino
1.2GHz	16 MHz
Build in Wi-Fi	No Wi-Fi
Single Board Computer	Microcontroller

Table 2 Rasberry Pi vs Ardunio

3.3.2 Product Subsystem2: Traffic Light and Stop Sign

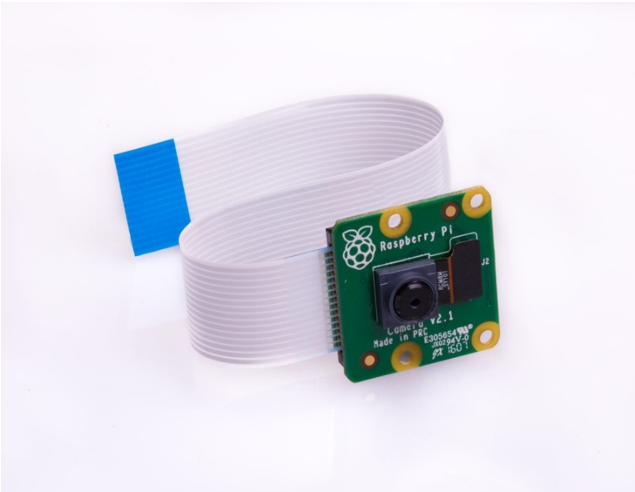


Figure 5 Pi Camera

The Raspberry Pi Camera Module is an official product from the Raspberry Pi Foundation. The original 5-megapixel model was released in 2013, and an 8-megapixel Camera Module v2 was released in 2016. For both iterations, there are visible light and infrared versions. We chose this camera for many reasons. First it is cheap and perfect for our application and it is easy to connect and use with the Raspberry Pi.

3.3.3 Product Subsystem3: Obstacle avoidance

Figure 6 Ultrasonic sensor

This subsystem is required to avoid road obstacles while the vehicle is travelling from the launch point to its destination. Since the car is driving at a low speed (180 m/h), the desired specifications of the sensor are not high. We set our design specifications requirements for the selected object detecting component to be having low detection range since our car will be driving at a low speed. Also, we desire the component to be consuming low power since the battery capacity will be limited.

3.4 Implementation

Python:

- Python is an interpreted high-level programming language for general-purpose programming.
- Python has a design philosophy that emphasizes code readability. It provides constructs that enable clear programming on both small and large scales.
- Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms and has a large and comprehensive standard library.

- Python interpreters are available for many operating systems.
- Python the reference implementation of Python is source software and has a community- based development model, as do nearly all of its variant implementations.

Opencv library: (Apache web server)

OpenCV is released under a BSD license and hence it's free for use. It has C++, Python and Java interfaces and supports Windows and Mac.

• OpenCV library is most widely used web server software.

• OpenCV library runs on 67% of all web servers in the world.

• OpenCV library is fast, reliable, and secure library.

• OpenCV library can be highly customized to meet the needs of many different environments by using extensions and modules.

NumPy library:

- NumPy library is the fundamental package for scientific computing with Python.
- NumPy library is a powerful N-dimensional array object.
- NumPy library is sophisticated (broadcasting) functions.
- NumPy library is tools for integrating C/C++ coding.

- NumPy library is useful linear algebra, Fourier transforms, and random number capabilities.

In order to drive the car using the PC and train it, we had to modify and hack the car transmitter. Our plan was to use Arduino and control the car directly. But we faced a problem to connect the rc car to the computer, so we modify the car transmitter by adding circuit made of relays and potentiometers then connect it between the Arduino and the transmitter in order to connect it to the computer. The figures below show the transmitter along with the Arduino and the circuit we made.

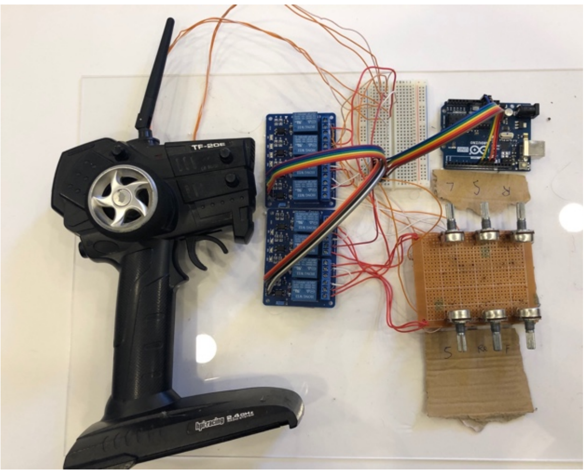


Figure 8 Modified Transmitter

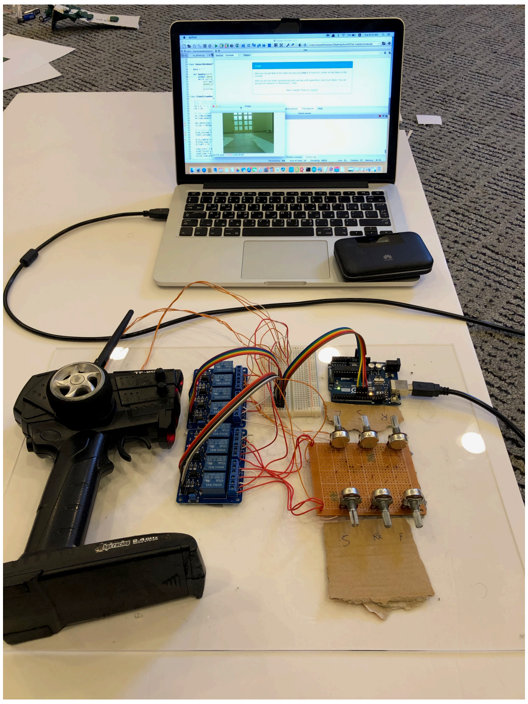


Figure 7 Modified Transmitter

4. System Testing and Analysis

4.1 Subsystem 1: Lane Following

The lane following main objective is to detect the lane borders. To do that we use a Raspberry Pi camera. The camera connected to the Raspberry Pi which transfer the image wirelessly via Wi-Fi to a laptop. The laptop will do the image processing and based on the results it will direct the car throw the lane.

The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction(steering), object detection (stop sign and traffic light), distance measurement (monocular vision), and sending instructions to Arduino through USB connection.

TCP Server

A multithread TCP server program runs on the computer to receive streamed image frames and ultrasonic data from the Raspberry Pi. Image frames are converted to gray scale and are decoded into NumPy arrays.

Figure 9 Lane following

Figure 10 Track

We tried to train the car to run between the lane in the above tracks, but unfortunately, we didn't get any positive result. Then we deiced to change the tracks to the following track below

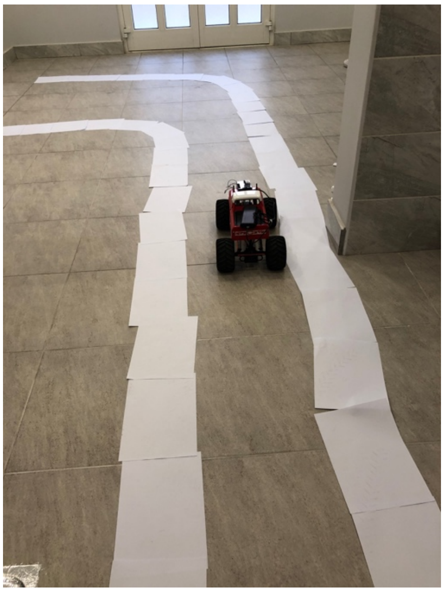


Figure 11 The car in the track

Neural Network

One advantage of using neural network is that once the network is trained, it only needs to load trained parameters afterwards, thus prediction can be very fast. Only lower half of the input image is used for training and prediction purposes. There are 38,400 (320×120) nodes in the input layer and 32 nodes in the hidden layer. The number of nodes in the hidden layer is chosen fairly arbitrary. There are four nodes in the output layer where each node corresponds to the steering control instructions: left, right, forward and reverse respectively (though reverse is not used anywhere in this project, it's still included in the output layer).

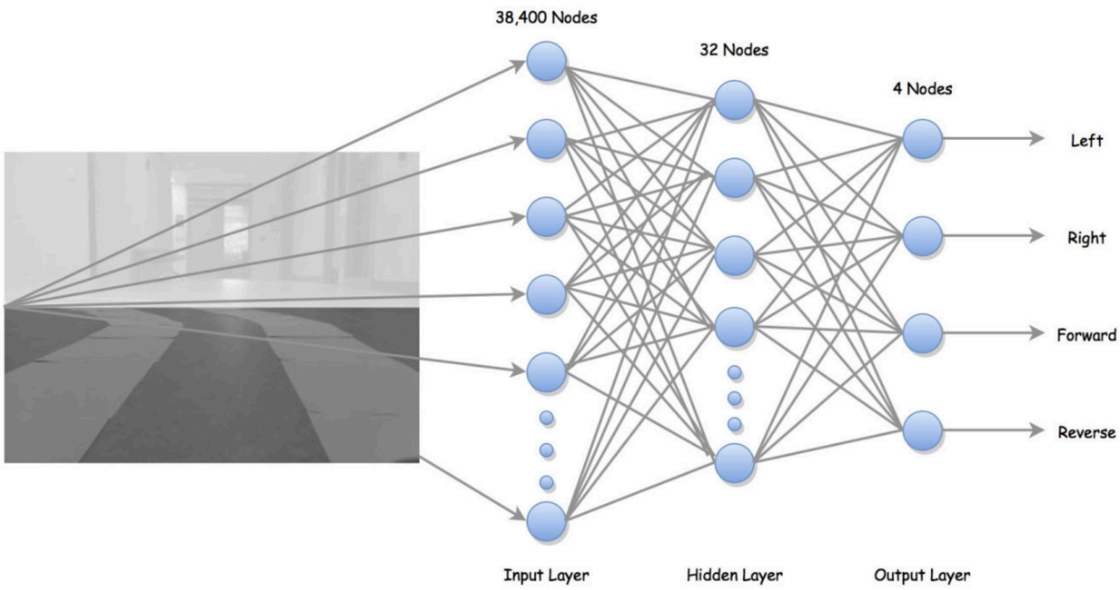


Figure 12 Neural Network

Below shows the training data collection process. First each frame is cropped and converted to a NumPy array. Then the train image is paired with train label (human input). Finally, all paired image data and labels are saved into a .npz file. The neural network is trained in OpenCV using back propagation method. Once training is done, weights are saved into a xml file. To generate predictions, the same neural network is constructed and loaded with the trained xml file.

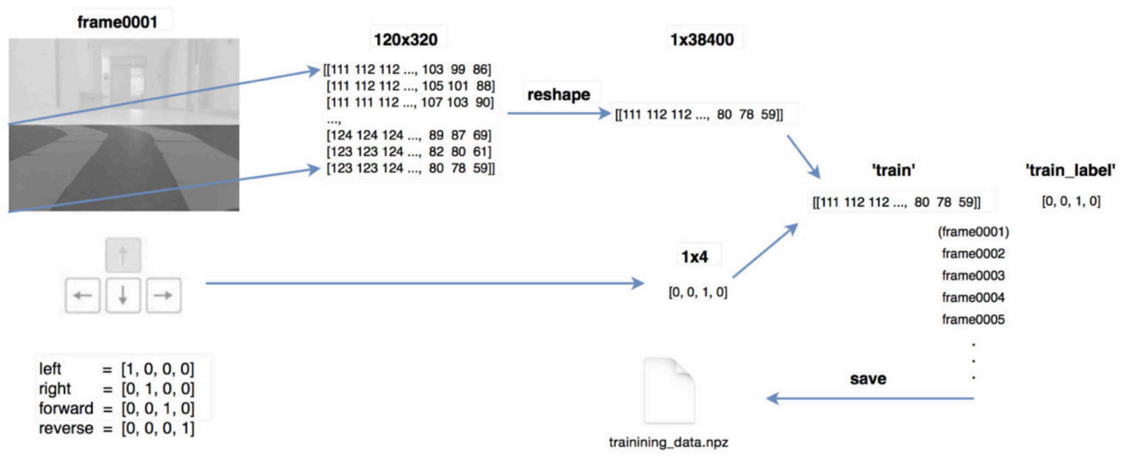


Figure 13 Neural Network

4.2 Subsystem 2: Traffic Light and Stop sign

The Traffic Light and Stop sign main objective is to detect the Traffic Lights and Stop signs. To do that we use a Raspberry Pi camera to identify them. The camera connected to the Raspberry Pi which transfer the image wirelessly via Wi-Fi to a laptop. The laptop will do the image processing and based on the results it will direct the car.

This project adapted the shape-based approach and used Haar feature-based cascade classifiers for object detection. Since each object requires its own classifier and follows the same process in training and detection, this project only focused on stop sign and traffic light detection.

OpenCV provides a trainer as well as detector. Positive samples (contain target object) were acquired using a cell phone and were cropped that only desired object is visible. Negative samples (without target object), on the other hand, were collected randomly. In particular, traffic light positive samples contain equal number of red traffic lights and green traffic light.

The same negative sample dataset was used for both stop sign and traffic light training. Below show some positive and negative samples used in this project.

Figure 14 Traffic Light and Stop Sign Samples

Type	Number of positive sample	Number of negative sample	Sample size (pixels)
Stop sign	20	400	25x25
Traffic light	26	400	25x45

Figure 15 Positive and negative samples

To recognize different states of the traffic light(red, green), some image processing is needed beyond detection. Flowchart below summarizes the traffic light recognition process.

Figure 16 Flowchart

Firstly, trained cascade classifier is used to detect traffic light. The bounding box is considered as a region of interest (ROI). Secondly, Gaussian blur is applied inside the ROI to reduce noises. Thirdly, find the brightest point in the ROI. Finally, red or green states are determined simply based on the position of the brightest spot in the ROI.

Distance Measurement

Raspberry Pi can only support one pi camera module. Using two USB web cameras will bring extra weight to the RC car and also seems unpractical. Therefore, monocular vision method is chosen.

This project adapted a geometry model of detecting distance to an object using monocular vision method proposed by Chu, Ji, Guo, Li and Wang (2004).

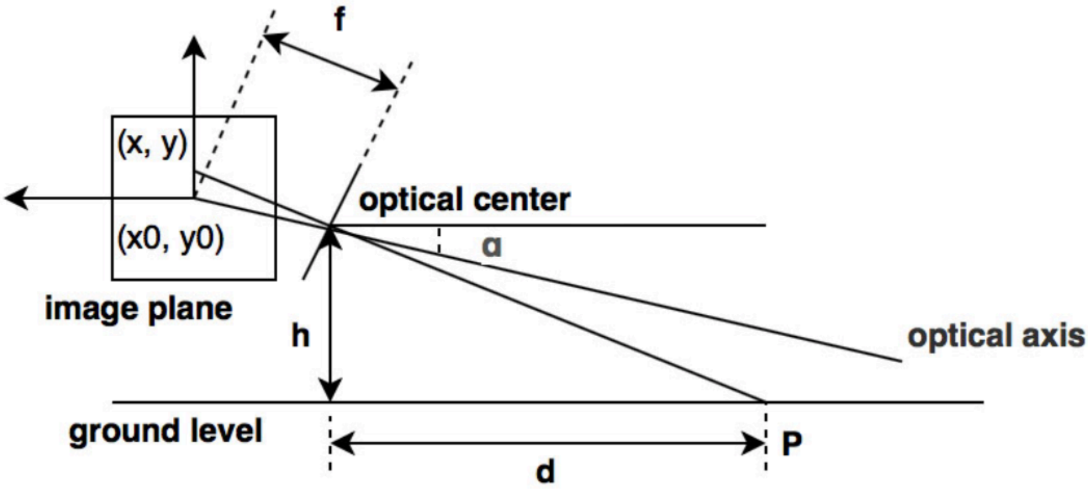


Figure 17 Distance measurements

P is a point on the target object; d is the distance from optical center to the point P. Based on the geometry relationship above, formula (1) shows how to calculate the distance d. In the formula (1), f is the focal length of the camera; ∂ is camera tilt angle; h is optical center height; (x0, y0) refers to the intersection point of image plane and optical axis; (x, y) refers to projection of point P on the image plane. Suppose O1 (u0,v0) is the camera coordinate of intersection point of optical axis and image plane, also suppose the physical dimension of a pixel corresponding to x-axis and y-axis on the image plane are dx and dy. Then:

$$d = h/\tan\left(\partial + \arctan\left((y - y_0)/f\right)\right) \tag{1}$$

$$u = \frac{x}{dx} + u_0 \quad v = \frac{y}{dy} + v_0 \tag{2}$$

Let $x_0 = y_0 = 0$, from (1) and (2):

$$d = h/\tan\left(\alpha + \arctan\left((v - v_0)/a_y\right)\right), \qquad (a_y = f/dy) \tag{3}$$

v is the camera coordinates on y-axis and can be returned from the object detection process. All other parameters are camera's intrinsic parameters that can be retrieved from camera matrix.

OpenCV provides functions for camera calibration. Camera matrix for the 5MP pi camera is returned after calibration. Ideally, a_x and a_y has the same value. Variance of these two values will result in non-square pixels in the image. The matrix below indicates that the fixed focal length lens on pi camera provides a reasonably good result in handling distortion aspect. Here is an interesting article discussing the focal length of pi camera with stock lens and its equivalent to 35mm camera.

$$\begin{bmatrix} a_x = 331.7 & 0 & u_0 = 161.9 \\ 0 & a_y = 332.3 & v_0 = 119.8 \\ 0 & 0 & 1 \end{bmatrix}$$

The matrix returns values in pixels and h is measured in centimeters. By applying formula (3), the physical distance d is calculated in centimeters.

Figure 18 Traffic Light detecting

Figure 19 Green Light Detecting

The following figures showing the stop sign detecting

Figure 20 Stop Sign

Figure 21 Stop Sign

4.3 Subsystem 3: Obstacle Detection

The Obstacle Detection main objective is to detect the obstacles and stop when there is. To do that we use an Ultrasonic sensor. The Sensor connected to the Raspberry Pi which transfer the data wirelessly via Wi-Fi to a laptop. The laptop will be deiced when the car stops based on the results.

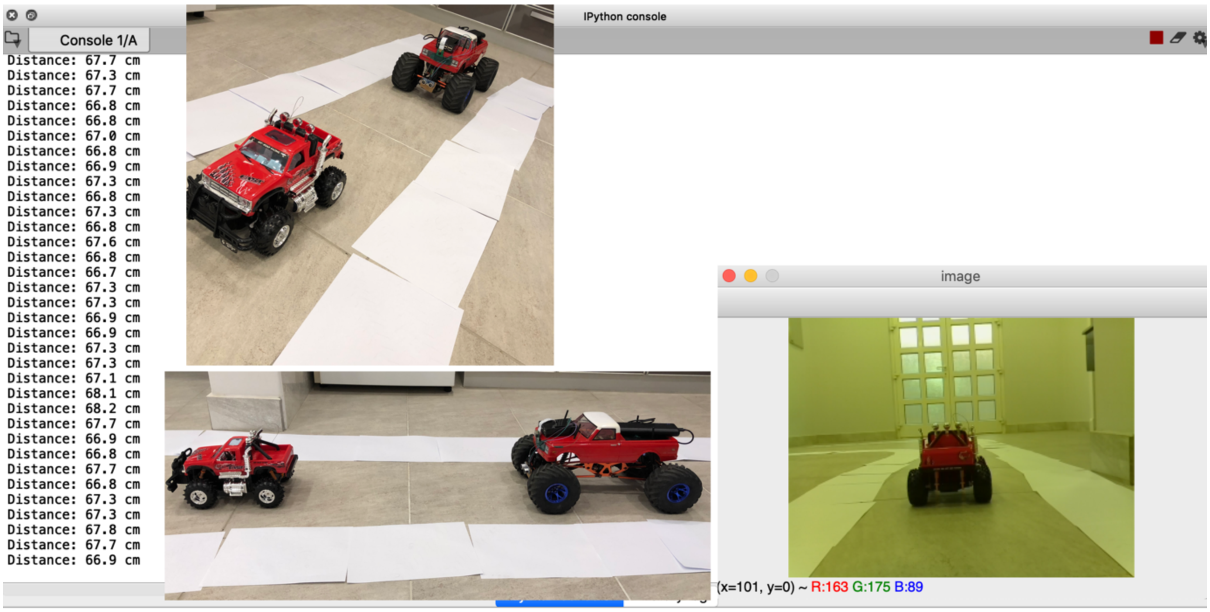


Figure 22 obstacle detecting

Figure 23 Obstacle detecting

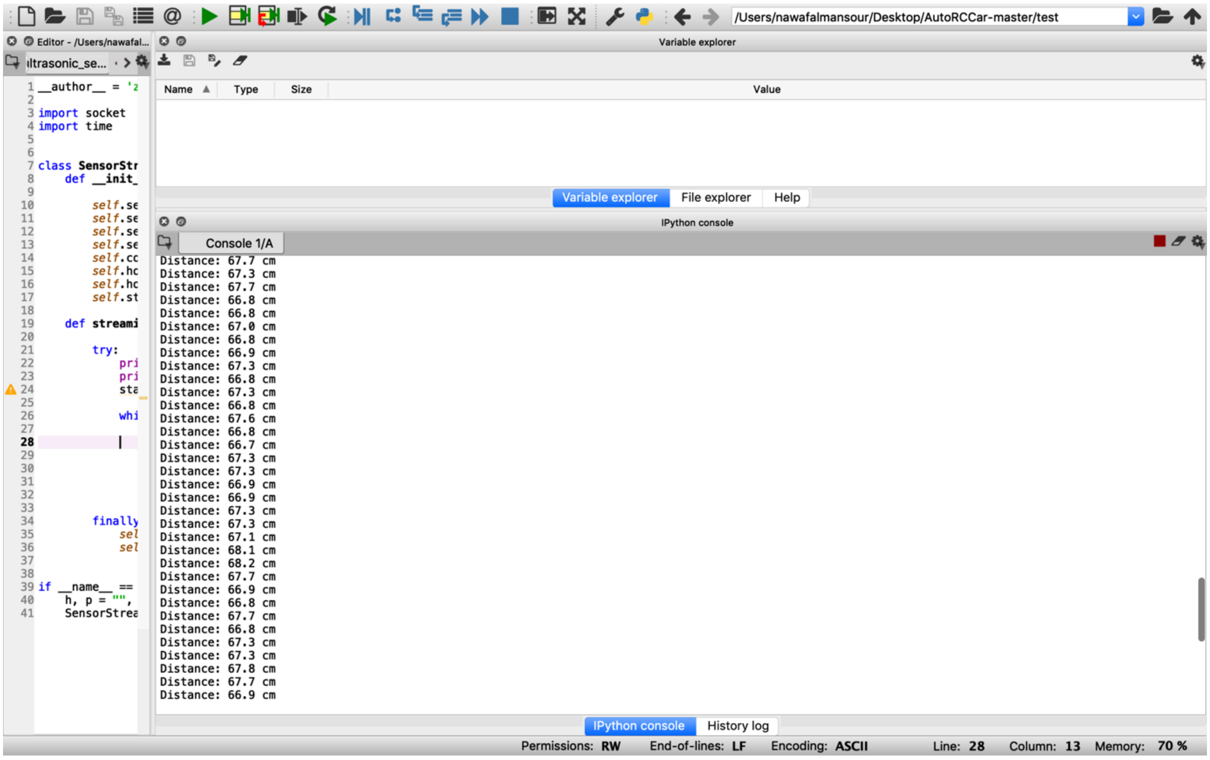


Figure 24 data read from the ultrasonic

4.4 Overall Results, Analysis and Discussion

Prediction on the testing samples returns an accuracy of 85% compared to the accuracy of 96% that the training samples returns. In actual driving situation, predictions are generated about 10 times a second (streaming rate roughly 10 frames/s).

Haar features by nature are rotation sensitive. In this project, however, rotation is not a concern as both the stop sign and the traffic light are fixed objects, which is also a general case in real world environment.

Figure 25 Traffic Light

Figure 26 Stop Sign

For distance measurement aspect, the ultrasonic sensor is only used to determine the distance to an obstacle in front of the RC car and provides accurate results when taking proper sensing angle and surface condition into considerations. On the other hand, Pi camera provides “good enough” measurement results. In fact, as long as we know the corresponding number to the actual distance, we know when to stop the RC car. Experimental results of detecting distance using pi camera are shown as below:

Order	1	2	3	4	5	6	7
Actual value (cm)	15	20	25	30	35	40	45
Measured value (cm)	15.5	19.7	24.1	27.5	32.0	35.2	39.0

Figure 27 measurements testing

In this project, the accuracy of distance measurement using monocular vision approach could be influenced by the following factors: (1) errors in actual values measurement, (2) object bounding box variations in detecting process, (3) errors in camera calibration process, (4) nonlinear relationship between distance and camera coordinate: the further the distance, the more rapid change of camera coordinate, thus the greater the error.

Overall, the RC car could successfully stop on the track to avoid front collision and respond to stop sign and traffic light accordingly. And for lane following unfortunately we couldn't succeed.

5. Project Management

5.1 Project Plan

During the design of this project we did the following:

1.1 We breakdown the tasks in a list and identify the tasks and the requirements.

1.2 We are collecting the date about the project to read and understand it

1.3 We map tasks and responsibilities to all team members and clarify and understand all the tasks and responsibilities.

5.2 Contribution of Team Members

Table 3 : Contribution of Team Members

Task	Hassan	Mahdi	Atif	Nawaf	Abdulhadi	Task Total
Search & acquire components	**30%**	**20%**	**20%**	**15%**	**15%**	**100%**
Design Subsystems	**25%**	**20%**	**15%**	**20%**	**20%**	**100%**
Test Subsystems	**20%**	**15%**	**15%**	**25%**	**25%**	**100%**
Write Reports & Presentations	**25%**	**25%**	**25%**	**15%**	**10%**	**100%**

5.3 Project Execution Monitoring

The team spent time all day together at PMU campus, whenever time was at hand we worked on the project. However, most of the time, we worked at night in the project room or sometime at any team member home.

Usually, if a team member encountered a problem, the other team members would provide back up and help in finishing that specific task. In addition, different ideas where discussed among the team members. Many of those ideas where agreed on, and some were rejected by the majority of the team members depending on the feasibility of the idea.

5.4 Challenges and Decision Making

1. Remote access a Raspberry PI from anther computer with application server
2. Automatically connect a raspberry pi to a Wi-Fi network
3. Synchronize all systems and time delay constraint
4. Batteries consumption was too high where we were forced to continuously charge them
5. Complexity of the coding of each component
6. Lacking the high-quality components which gives much better performance

5.5 Project Bill of Materials and Budget

Item	Quantity	Unit Cost (SR)
Microcontroller	2	200
Pi Camera	1	90
DC Battery	2	75
DC Motor	1	170
Chassis	1	240
ESC	1	250
Transmitter & Receiver	1	450
Servo	1	120
	Total	1870

6. Project Analysis

6.1 Life-long Learning

6.1.1 WHAT ARE ARTIFICIAL NEURAL NETWORKS?

Artificial neural networks (ANNs) are software implementations of the neuronal structure of our brains. We don’t need to talk about the complex biology of our brain structures, but suffice to say, the brain contains neurons which are kind of like organic switches. These can change their output state depending on the strength of their electrical or chemical input. The neural network in a person’s brain is a hugely interconnected network of neurons, where the output of any given neuron may be the input to thousands of other neurons. Learning occurs by repeatedly activating certain neural connections over others, and this reinforces those connections. This makes them more likely to produce a desired outcome given a specified input. This learning involves feedback – when the desired outcome occurs, the neural connections causing that outcome becomes strengthened. Artificial neural networks attempt to simplify and mimic this brain behavior. They can be trained in a supervised or unsupervised manner. In a supervised ANN, the network is trained by providing matched input and output data samples, with the intention of getting the ANN to provide a desired output for a given input. An example is an e-mail spam filter – the input training data could be the count of various words in the body of the email, and the output training data would be a classification of whether the e-mail was truly spam or not. If many examples of e-mails are passed through the neural network this allows the network to learn what input data makes it likely that an e-mail is spam or not. This learning takes place be adjusting the weights of the ANN connections, but this will be discussed further in the next section. Unsupervised learning in an ANN is an attempt to get the ANN to “understand” the structure of the provided input data “on its own”.

6.1.2 The Structure of an ANN

6.1.2.1 The artificial neuron

is simulated in an ANN by an activation function. In classification tasks (e.g. identifying spam e-mails) this activation function must have a "switch on" characteristic – in other words, once the input is greater than a certain value, the output should change state i.e. from 0 to 1, from -1 to 1 or from 0 to >0. This simulates the "turning on" of a biological neuron. A common activation function that is used is the sigmoid function:

$$F(z) = 1/(1+exp(-z))$$

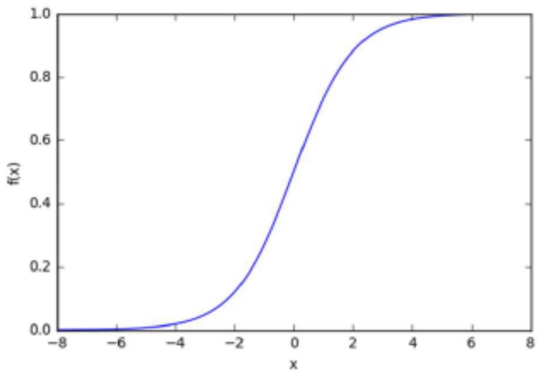


Figure 28 The sigmoid function

As can be seen in the figure above, the function is "activated" i.e. it moves from 0 to 1 when the input x is greater than a certain value. The sigmoid function isn't a step function however, the edge is "soft", and the output doesn't change instantaneously. This means that there is a derivative of the function and this is important for the training algorithm.

6.1.2.2 Nodes

As mentioned previously, biological neurons are connected hierarchical networks, with the outputs of some neurons being the inputs to others. We can represent these networks as connected layers of nodes. Each node takes multiple weighted inputs, applies the activation function to the summation of these inputs, and in doing so generates an output. I'll break this down further, but to help things along, consider the diagram below:

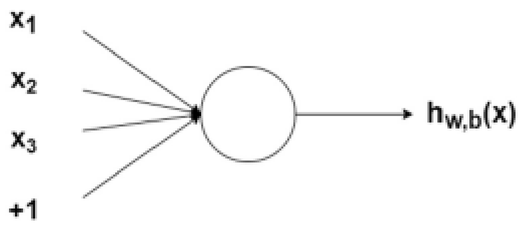


Figure 29 Node with inputs

The circle in the image above represents the node. The node is the "seat" of the activation function, and takes the weighted inputs, sums them, then inputs them to the activation function. The output of the activation function is shown as h in the above diagram. Note: a node as I have shown above is also called a perceptron in some literature. What about this "weight" idea that has been mentioned? The weights are real valued numbers (i.e. not binary 1s or 0s), which are multiplied by the inputs and then summed up in the node. So, in other words, the weighted input to the node above would be:

$$x_1w_1 + x_2w_2 + x_3w_3 + b$$

Here the w_i values are weights. What are these weights all about? Well, they are the variables that are changed during the learning process, and, along with the input, determine the output of the node. The b is the weight of the +1-bias element – the inclusion of this bias enhances the flexibility of the node, which is best demonstrated in an example.

Let’s take an extremely simple node, with only one input and one output:

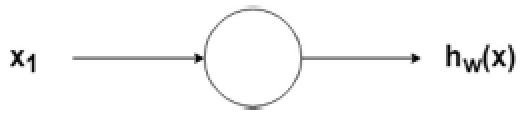


Figure 30 Simple node

Putting together the structure

Hopefully the previous explanations have given a good overview of how a given node/neuron/perceptron in a neural network operates. However, as you are probably aware, there are many such interconnected nodes in a fully-fledged neural network. These structures can come in a myriad of different forms, but the most common simple neural network structure consists of an input layer, a hidden layer and an output layer. An example of such a structure can be seen below:

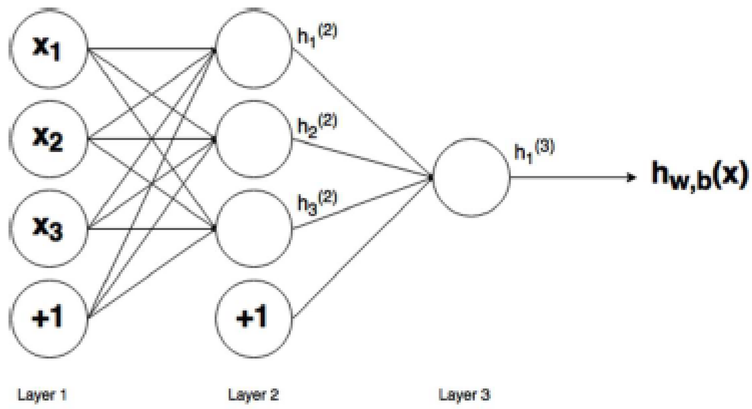


Figure 31 Three-layer neural network

The three layers of the network can be seen in the above figure – Layer 1 represents the input layer, where the external input data enters the network. Layer 2 is called the hidden layer as this layer is not part of the input or output. Note: neural networks can have many hidden layers, but in this case for simplicity I have just included one. Finally, Layer 3 is the output layer. You can observe the many connections between the layers, in particular between Layer 1 (L1) and Layer 2 (L2). As can be seen, each node in L1 has a connection to all the nodes in L2. Likewise, for the nodes in L2 to the single output node L3. Each of these connections will have an associated weight.

6.2 Impact of Engineering Solutions

The smart car can be a very good solution for a lot of problems and its impact can be noticed in environment, society and economy. It can help the handicapped people and make them more affective in the society. In addition, the project can help the environment by reducing the gas emissions. The gas emissions produced by normal cars are very harmful. Also, it will become more eco friendly if the car uses renewable energy. Moving to the economical effects. The smart car can cost less in the fuel cost and it will be cheapest if it uses the renewable energy. This technology is expensive, but it will decrease in the future.

7. Conclusions and Future Recommendations

7.1 Conclusions

The main purpose of the project was to Smart Electrical Car which we have Semi-successfully accomplished. This has been a great learning experience for the team, and we are very satisfied with our efforts. Autonomous is a promising technology that is expected to have a number of significant benefits to society: increased mobility, better utilization of lands, reduced costs of congestion or increased road efficiency, and dramatically decreased car accidents. This technology is still in its early stage and is far from being fully autonomous with several technical challenges yet to be overcome.

7.2 Future Recommendations

The project can be enhanced and improved from many perspectives. Below are two recommendations for future:

- Install Camera in the car back for reserving direction This will help the car inside the parking area, in this case car can parked autonomous Higher torque DC servo motors
- Install higher mechanical tropic DC servo motors for next level speedy Car
- Larger scale car for transporting human

8. References

Helmers, E., & Marx, P. (2012). Electric cars: technical characteristics and environmental impacts. *Environmental Sciences Europe*, *24*(1), 14.

Lassila, J., Haakana, J., Tikka, V., & Partanen, J. (2012). Methodology to analyze the economic effects of electric cars as energy storages. *IEEE Transactions on smart grid*, *3*(1), 506-516.

Musselwhite, C., & Haddad, H. (2010). Mobility, accessibility and quality of later life. *Quality in Ageing and Older Adults*, *11*(1), 25-37.

Van Vliet, O., Brouwer, A. S., Kuramochi, T., van Den Broek, M., & Faaij, A. (2011). Energy use, cost and CO2 emissions of electric cars. *Journal of power sources*, *196*(4), 2298-2310.

https://static.googleusercontent.com/media/www.google.com/en//selfdrivingcar/files/reports/report-1116.pdf

https://spectrum.ieee.org/static/the-self-driving-car

https://techcrunch.com/2017/09/27/luminar-lidar-toyota-research-institute/

https://www.google.com/selfdrivingcar/

https://www.kdnuggets.com/2017/06/machine-learning-algorithms-used-self-driving-

https://en.wikipedia.org/wiki/Autonomous_car

https://www.theguardian.com/technology/self-driving-cars

https://www.engadget.com/2017/12/27/lg-and-here-team-on-self-driving-car-hub/

https://store.arduino.cc/usa/arduino-mega-2560-rev3

https://www.arduino.cc/

https://github.com/hamuchiwa/AutoRCCar

Appendix A: Progress Reports

Smart Electrical Car

Nawaf Almansour 201301171
Hassan Alghamdi 201301928
Atif Alqahtani 201402544
Mahdi Alfarwan 201002050
Abdulhadi Alhajri 201500089

Design II (ASSE 3) | **Spring 2019**

ProgRpt No. 1

Instructor: Dr. Chedly B. Yahya

Period Highlight: 1 | Plan | Actual | Actual (beyond plan) | % Complete (beyond plan)

Periods (Weeks 1-15): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ACTIVITY	PLAN START	PLAN DURATION	Assigned To	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Write plan	1	1	ALL	2	2	100%
Design Subsystem 2 (Obstacle Avoida	2	1	AH,MF	2	2	40%
Test subsystem 1 (Lane Following)	2	3	NM,HG	2	2	30%
Verfiy & refine Subsys 1	3	2	ALL	3	2	0%
Write Progress Report	3	1	ALL	3	4	0%
Testing Car motor and steering	4	1	HG,MF	4	4	65%
Design communication system	5	2	NM,AQ	5	4	0%
Testing communication system	5	1	AH,NM	5	3	0%
Applying subsystem 1 on the car	5	1	AQ,MF	5	3	0%
Calibrate subsys1 on car if needed	6	1	NM,HG	6	5	0%
Design Weather system	6	1	ALL	5	2	0%
Prepare midterm Presentation	7	2	ALL	7	3	50%
Apply Weather system & test it	8	2	AH,AQ	8	2	0%
Design subsystem 3 (Traffic Light)	9	2	ALL	8	4	0%
Test subsystem 3	10	1	NM,HG	10	2	0%
Intgerate all Subsystems	11	2	ALL	12	2	0%
Prepapre final report	12	2	ALL	12	2	25%
Prepapre final presentation	12	2	ALL	12	3	0%
Prepare Project Demo	13	2	ALL	13	3	0%
Submit Rpt/PPT/Brochure	14	2	ALL	14	2	0%

Title: Smart Electric Car | **Advisor: Dr. Chedly B. Yahya**

Nawaf Almansour 201301171
Hassan Alghamdi 201301928
Atif Alqahtani 201402544
Mahdi Alfarwan 201002050
Abdulhadi Alhajri 201500089

Design II (ASSE 3) | **Spring 2019**

Project PLAN & Progress

ProgRpt No. 2

Plan updated (Date): Feb 3, 2019

Instructor: Dr. Chedly B. Yahya

Period Highlight: 4 | Plan | Actual | Actual (beyond plan) | % Complete (beyond plan)

Periods (Weeks 1-15): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ACTIVITY	PLAN START	PLAN DURATION	Assigned To	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Write plan	1	1	ALL	2	2	100%
Design Subsystem 2 (Obstacle Avoidance)	2	1	AH,MF	2	2	50%
Test subsystem 1 (Lane Following)	2	3	NM,HG	4	3	70%
Verfiy & refine Subsys 1	3	2	ALL	3	2	0%
Write Progress Report	3	1	ALL	5	1	30%
Testing Car motor and steering	4	1	HG,MF	4	4	40%
Design communication system	5	2	NM,AQ			0%
Testing communication system	5	1	AH,NM			0%
Applying subsystem 1 on the car	5	1	AQ,MF			0%
Calibrate subsys1 on car if needed	6	1	NM,HG			0%
Design Weather system	6	1	ALL			0%
Prepare midterm Presentation	7	2	ALL			50%
Apply Weather system & test it	8	2	AH,AQ			0%
Design subsystem 3 (Traffic Light)	9	2	ALL			0%
Test subsystem 3	10	1	NM,HG			0%
Intgerate all Subsystems	11	2	ALL			0%
Prepapre final report	12	2	ALL			25%
Prepapre final presentation	12	2	ALL			0%
Prepare Project Demo	13	2	ALL			0%
Submit Rpt/PPT/Brochure	14	2	ALL			0%
						0%
						0%

Progress Details:

The motor we ordered will be delivered in Feb. 13, 2019 - Feb. 18, 2019

We were able to run parts of the code and transfer the images to the labtop

Issues (delay ...):

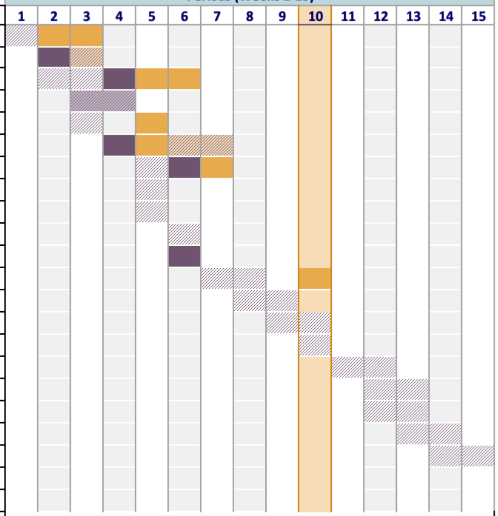
We had some problems with the motor. So we ordered new one

We had some missing part in the last testing for the lane follower (Arduino)

Title: Smart Electric Car	Advisor: Dr. Chedly B. Yahya	Design II (ASSE 3)	Spring 2019
Nawaf Almansour 201301171		Project PLAN & Progress	
Hassan Alghamdi 201301928		ProgRpt No. 3	
Atif Alqahtani 201402544		Plan updated (Date): Mar 10, 2019	
Mahdi Alfarwan 201002050		Instructor: Dr. Chedly B. Yahya	
Abdulhadi Alhajri 201500089		Period Highlight: 10	Plan / Actual
		Actual (beyond plan)	% Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	Assigned To	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Write plan	1	1	ALL	2	2	100%
Design Subsystem 2 (Obstacle Avoidance)	2	1	AH,MF	2	2	50%
Test subsystem 1 (Lane Following)	2	3	NM,HG	4	3	100%
Verfiy & refine Subsys 1	3	2	ALL	3	2	0%
Write Progress Report	3	1	ALL	5	1	30%
Testing Car motor and steering	4	1	HG,MF	4	4	60%
Design communication system	5	2	NM,AQ	6	2	100%
Testing communication system	5	1	AH,NM			0%
Applying subsystem 1 on the car	5	1	AQ,MF			0%
Calibrate subsys1 on car if needed	6	1	NM,HG			0%
Design Weather system	6	1	ALL	6	1	40%
Prepare midterm Presentation	7	2	ALL	10	1	100%
Apply Weather system & test it	8	2	AH,AQ			0%
Design subsystem 3 (Traffic Light)	9	2	ALL			0%
Test subsystem 3	10	1	NM,HG			0%
Intgerate all Subsystems	11	2	ALL			0%
Prepapre final report	12	2	ALL			25%
Prepapre final presentation	12	2	ALL			25%
Prepare Project Demo	13	2	ALL			0%
Submit Rpt/PPT/Brochure	14	2	ALL			0%
						0%
						0%

Periods (Weeks 1-15)



Progress Details:

we are done with the lane following subsystem

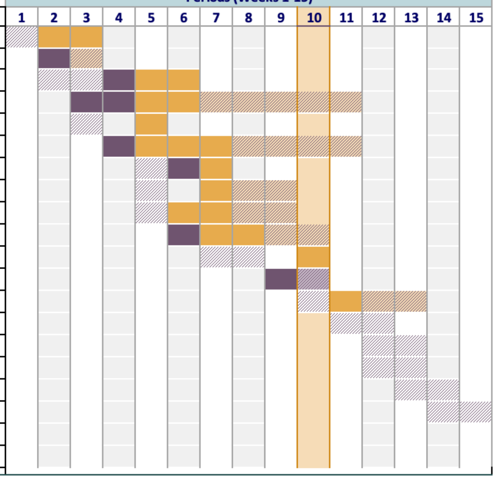
Issues (delay ...):

We having some problems with the car which cause delay

Title: Smart Electric Car	Advisor: Dr. Chedly B. Yahya	Design II (ASSE 3)	Spring 2019
Nawaf Almansour 201301171		Project PLAN & Progress	
Hassan Alghamdi 201301928		ProgRpt No. 4	
Atif Alqahtani 201402544		Plan updated (Date): Mar 25, 2019	
Mahdi Alfarwan 201002050		Instructor: Dr. Chedly B. Yahya	
Abdulhadi Alhajri 201500089		Period Highlight: 10	Plan / Actual
		Actual (beyond plan)	% Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	Assigned To	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Write plan	1	1	ALL	2	2	100%
Design Subsystem 2 (Obstacle Avoidance)	2	1	AH,MF	2	2	50%
Test subsystem 1 (Lane Following)	2	3	NM,HG	4	3	100%
Verfiy & refine Subsys 1	3	2	ALL	3	9	45%
Write Progress Report	3	1	ALL	5	1	100%
Testing Car motor and steering	4	1	HG,MF	4	8	60%
Design communication system	5	2	NM,AQ	6	2	100%
Testing communication system	5	1	AH,NM	7	3	50%
Applying subsystem 1 on the car	5	1	AQ,MF	6	4	55%
Calibrate subsys1 on car if needed	6	1	NM,HG	6	5	60%
Prepare midterm Presentation	7	2	ALL	10	1	100%
Design subsystem 3 (Traffic Light)	9	2	ALL	9	2	44%
Test subsystem 3	10	1	NM,HG	11	3	35%
Intgerate all Subsystems	11	2	ALL			0%
Prepapre final report	12	2	ALL			25%
Prepapre final presentation	12	2	ALL			25%
Prepare Project Demo	13	2	ALL			20%
Submit Rpt/PPT/Brochure	14	2	ALL			0%
						0%
						0%

Periods (Weeks 1-15)

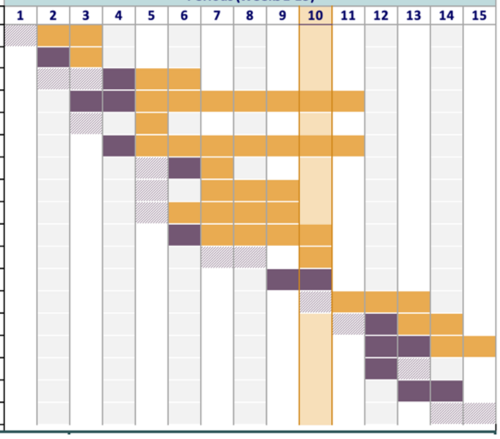


Title: Smart Electric Car	Advisor: Dr. Chedly B. Yahya	Design II (ASSE 3)	Spring 2019
Nawaf Almansour 201301171		Project PLAN & Progress	
Hassan Alghamdi 201301928		ProgRpt No. 5	
Atif Alqahtani 201402544		Plan updated (Date): Apr 20, 2019	
Mahdi Alfarwan 201002050		Instructor: Dr. Chedly B. Yahya	
Abdulhadi Alhajri 201500089			

Period Highlight: 10 | Plan | Actual | Actual (beyond plan) | % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	Assigned To	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Write plan	1	1	ALL	2	2	100%
Design Subsystem 2 (Obstacle Avoidance)	2	1	AH,MF	2	2	100%
Test subsystem 1 (Lane Following)	2	3	NM,HG	4	3	100%
Verfiy & refine Subsys 1	3	2	ALL	3	9	100%
Write Progress Report	3	1	ALL	5	1	100%
Testing Car motor and steering	4	1	HG,MF	4	8	100%
Design communication system	5	2	NM,AQ	6	2	100%
Testing communication system	5	1	AH,NM	7	3	100%
Applying subsystem 1 on the car	5	1	AQ,MF	6	4	100%
Calibrate subsys1 on car if needed	6	1	NM,HG	6	5	100%
Prepare midterm Presentation	7	2	ALL	10	1	100%
Design subsystem 3 (Traffic Light)	9	2	ALL	9	2	100%
Test subsystem 3	10	1	NM,HG	11	3	100%
Intgerate all Subsystems	11	2	ALL	12	3	100%
Prepapre final report	12	2	ALL	12	4	100%
Prepapre final presentation	12	2	ALL	12	1	100%
Prepare Project Demo	13	2	ALL	13	2	100%
Submit Rpt/PPT/Brochure	14	2	ALL			100%

Periods (Weeks 1-15)



Appendix B: Bill of Materials

Item	Quantity	Unit Cost (SR)	Subtotal
Microcontroller	2	200	400
Pi Camera	1	90	90
DC Battery	2	75	150
DC Motor	1	170	170
Chassis	1	240	240
ESC	1	250	250
Transmitter & Receiver	1	450	450
Servo	1	120	120
		Total	1870

Appendix C: Datasheets

Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

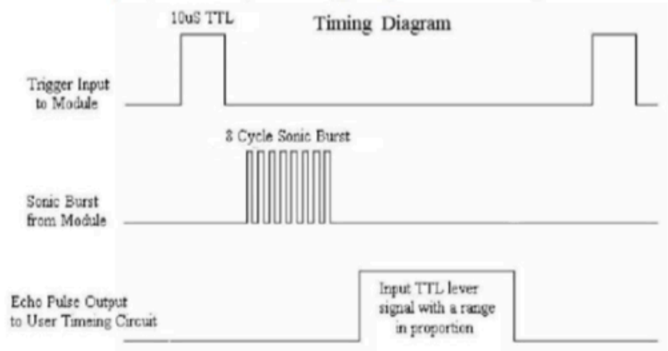
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

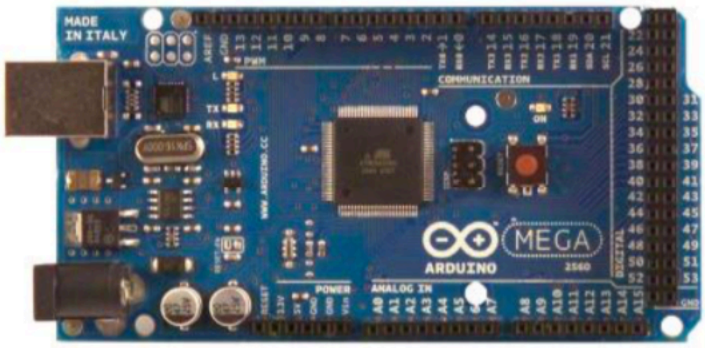
Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

RS radiospares RADIONICS

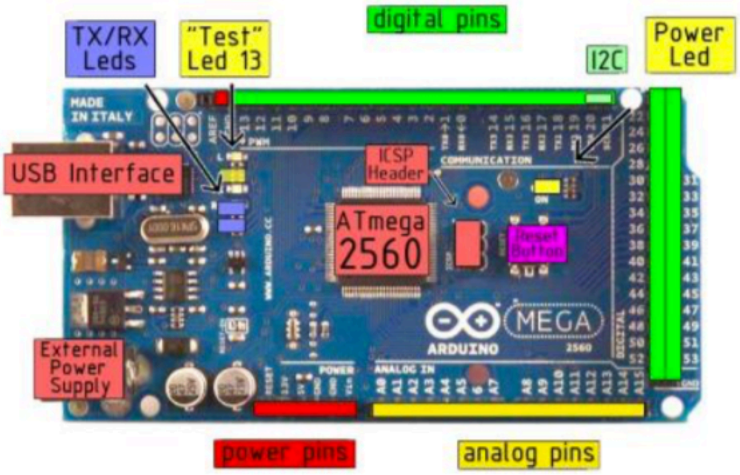
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

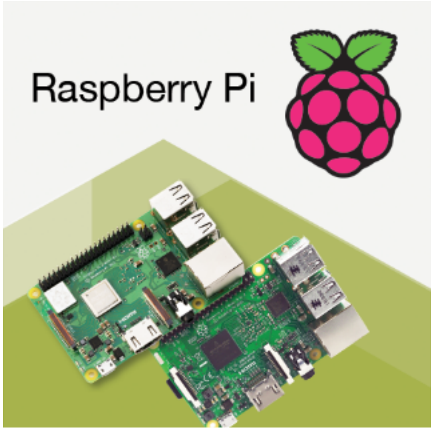
Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



RS radiospares RADIONICS



6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

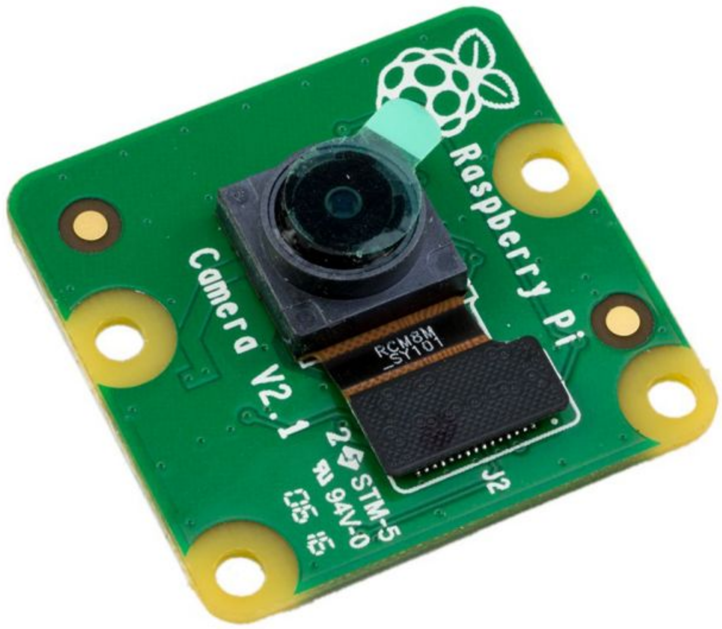
Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

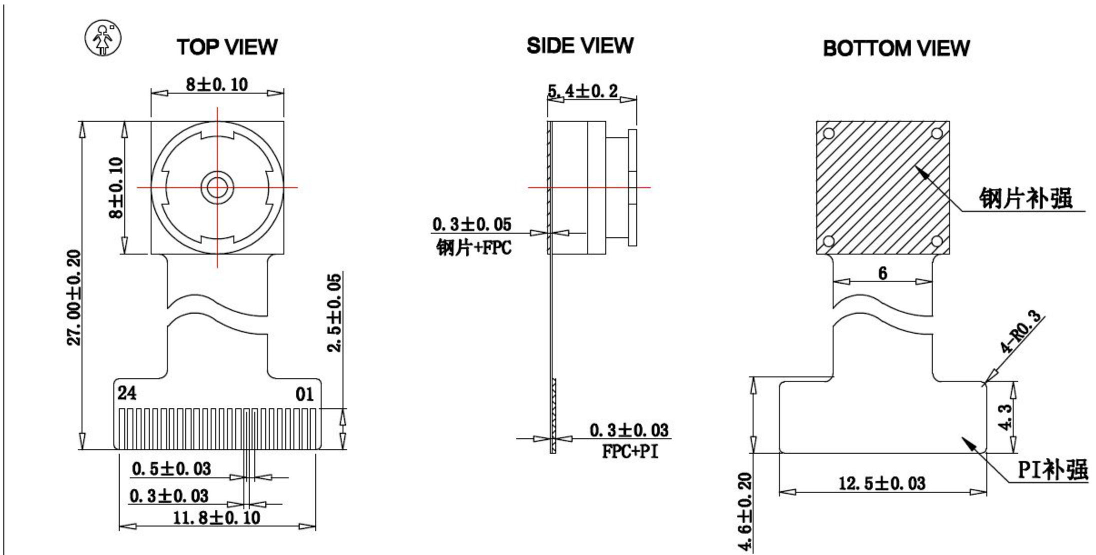
Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)





Lens Specification

Optical Specification	
MAX IMAGE CIRCLE	
Focal Length	
FOV (D)	62°
TV Distortion	<1%
F .Number	2.8

PIN DEFINITION

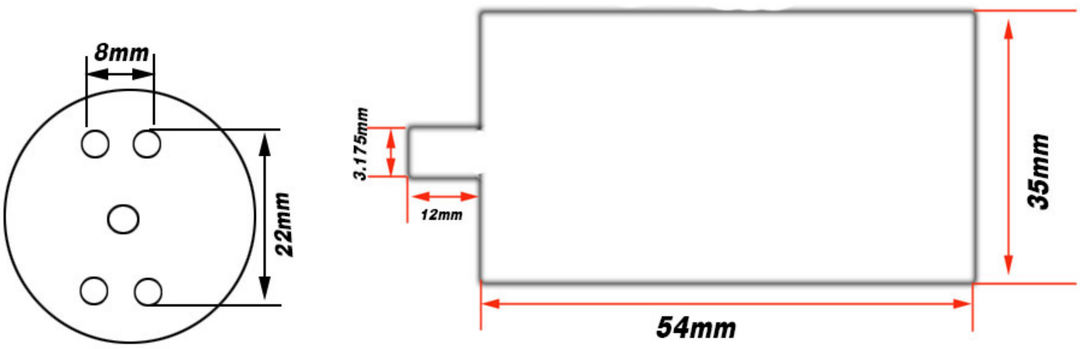
PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	NC	6	RESET	11	DOVDD2V8	16	D5	21	D1
2	AGND	7	VSYNC	12	D7	17	PCLK	22	D2
3	SIO_D	8	PWDN	13	MCLK	18	D4	23	NC
4	AVDD2V8	9	HREF	14	D6	19	D0	24	NC
5	SIO_C	10	DVDD1V8	15	DGND	20	D3		

NOTE

1 线性尺寸未注公差均为±0.1mm
2 Sensor:OV7670
3 Image direction register use default settings.

4				一般公差	
3				尺寸范围	容许公差
2				ANGLE	±0.5'
1	初稿		13.03.19		
NO.	更改	NAME	DATE		

				SCALE		PROJ.	
				REV	1	QTY	1
TITLE	CMOS Camera Module			SHEET	1 OF 1	SIZE	A4
DRAWING		DATE	13.03.19	Part Name	HD-7670-01		
CHECKED		UNIT	MM	Customer P/N			
APPROVED		WEIGHT		Material			



Goolrc 540 car motor

- 7.4 Volt
- Max current: 1.82A
- 16500 RPM
- Shaft length: 12mm

Appendix D: Program Codes

Collect training data code

```
import
numpy
as np
        import cv2
        import serial
        import pygame
        from pygame.locals import *
        import socket
        import time
        import os


        class CollectTrainingData(object):

            def __init__(self, host, port, serial_port, input_size):

                self.server_socket = socket.socket()
                self.server_socket.bind((host, port))
                self.server_socket.listen(0)


                self.connection = self.server_socket.accept()[0].makefile('rb')

                self.ser = serial.Serial(serial_port, 115200, timeout=1)
                self.send_inst = True

                self.input_size = input_size


                self.k = np.zeros((4, 4), 'float')
                for i in range(4):
                    self.k[i, i] = 1

                pygame.init()
                pygame.display.set_mode((250, 250))

            def collect(self):

                saved_frame = 0
                total_frame = 0
```

```
        print("Start collecting images...")
        print("Press 'q' or 'x' to finish...")
        start = cv2.getTickCount()

        X = np.empty((0, self.input_size))
        y = np.empty((0, 4))


        try:
            stream_bytes = b' '
            frame = 1
            while self.send_inst:
                stream_bytes += self.connection.read(1024)
                first = stream_bytes.find(b'\xff\xd8')
                last = stream_bytes.find(b'\xff\xd9')

                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]
                    image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
cv2.IMREAD_GRAYSCALE)


                    height, width = image.shape
                    roi = image[int(height/2):height, :]

                    cv2.imshow('image', image)



                    temp_array = roi.reshape(1, int(height/2) *
width).astype(np.float32)

                    frame += 1
                    total_frame += 1

                    for event in pygame.event.get():
                        if event.type == KEYDOWN:
                            key_input = pygame.key.get_pressed()

                            # complex orders
                            if key_input[pygame.K_UP] and
key_input[pygame.K_RIGHT]:
                                print("Forward Right")
                                X = np.vstack((X, temp_array))
```

```
                y = np.vstack((y, self.k[1]))
                saved_frame += 1
                self.ser.write(chr(6).encode())

            elif key_input[pygame.K_UP] and
key_input[pygame.K_LEFT]:
                print("Forward Left")
                X = np.vstack((X, temp_array))
                y = np.vstack((y, self.k[0]))
                saved_frame += 1
                self.ser.write(chr(7).encode())

            elif key_input[pygame.K_DOWN] and
key_input[pygame.K_RIGHT]:
                print("Reverse Right")
                self.ser.write(chr(8).encode())

            elif key_input[pygame.K_DOWN] and
key_input[pygame.K_LEFT]:
                print("Reverse Left")
                self.ser.write(chr(9).encode())

            elif key_input[pygame.K_UP]:
                print("Forward")
                saved_frame += 1
                X = np.vstack((X, temp_array))
                y = np.vstack((y, self.k[2]))
                self.ser.write(chr(1).encode())

            elif key_input[pygame.K_DOWN]:
                print("Reverse")
                self.ser.write(chr(2).encode())

            elif key_input[pygame.K_RIGHT]:
                print("Right")
                X = np.vstack((X, temp_array))
                y = np.vstack((y, self.k[1]))
                saved_frame += 1
                self.ser.write(chr(3).encode())

            elif key_input[pygame.K_LEFT]:
                print("Left")
                X = np.vstack((X, temp_array))
                y = np.vstack((y, self.k[0]))
                saved_frame += 1
                self.ser.write(chr(4).encode())
```

```
                            elif key_input[pygame.K_x] or
key_input[pygame.K_q]:
                                print("exit")
                                self.send_inst = False
                                self.ser.write(chr(0).encode())
                                self.ser.close()
                                break

                        elif event.type == pygame.KEYUP:
                            self.ser.write(chr(0).encode())

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break

            file_name = str(int(time.time()))
            directory = "training_data"
            if not os.path.exists(directory):
                os.makedirs(directory)
            try:
                np.savez(directory + '/' + file_name + '.npz', train=X,
train_labels=y)
            except IOError as e:
                print(e)

            end = cv2.getTickCount()

            print("Streaming duration: , %.2fs" % ((end - start) /
cv2.getTickFrequency()))

            print(X.shape)
            print(y.shape)
            print("Total frame: ", total_frame)
            print("Saved frame: ", saved_frame)
            print("Dropped frame: ", total_frame - saved_frame)

        finally:
            self.connection.close()
            self.server_socket.close()


if __name__ == '__main__':
    # host, port
    h, p = "192.168.1.100", 8000
```

```
sp = "/dev/tty.usbmodem1421"

s = 120 * 320

ctd = CollectTrainingData(h, p, sp, s)
ctd.collect()
```

model_training code:

```
from model
import
load_data,
NeuralNetwork

input_size = 120 * 320
data_path = "training_data/*.npz"

X_train, X_valid, y_train, y_valid = load_data(input_size, data_path)

# train a neural network
layer_sizes = [input_size, 32, 4]
nn = NeuralNetwork()
nn.create(layer_sizes)
nn.train(X_train, y_train)

# evaluate on train data
train_accuracy = nn.evaluate(X_train, y_train)
print("Train accuracy: ", "{0:.2f}%".format(train_accuracy * 100))

# evaluate on validation data
validation_accuracy = nn.evaluate(X_valid, y_valid)
print("Validation accuracy: ", "{0:.2f}%".format(validation_accuracy *
100))

# save model
model_path = "saved_model/nn_model.xml"
nn.save_model(model_path)
```

RC Driver code:

```
import
sys
        import threading
        import socketserver
        import numpy as np


        from model import NeuralNetwork
        from rc_driver_helper import *

        # distance data measured by ultrasonic sensor
        sensor_data = None


        class SensorDataHandler(socketserver.BaseRequestHandler):

            data = " "

            def handle(self):
                global sensor_data
                while self.data:
                    self.data = self.request.recv(1024)
                    sensor_data = round(float(self.data), 1)
                    # print "{} sent:".format(self.client_address[0])
                    print(sensor_data)


        class VideoStreamHandler(socketserver.StreamRequestHandler):

            # h1: stop sign, measured manually
            # h2: traffic light, measured manually
            h1 = 5.5  # cm
            h2 = 5.5

            # load trained neural network
            nn = NeuralNetwork()
            nn.load_model("saved_model/nn_model.xml")

            obj_detection = ObjectDetection()
            rc_car = RCControl("/dev/tty.usbmodem1421")

            # cascade classifiers
            stop_cascade = cv2.CascadeClassifier("cascade_xml/stop_sign.xml")
```

```
    light_cascade = cv2.CascadeClassifier("cascade_xml/traffic_light.xml")

    d_to_camera = DistanceToCamera()
    # hard coded thresholds for stopping, sensor 30cm, other two 25cm
    d_sensor_thresh = 30
    d_stop_light_thresh = 25
    d_stop_sign = d_stop_light_thresh
    d_light = d_stop_light_thresh

    stop_start = 0  # start time when stop at the stop sign
    stop_finish = 0
    stop_time = 0
    drive_time_after_stop = 0

    def handle(self):

        global sensor_data
        stream_bytes = b' '
        stop_flag = False
        stop_sign_active = True

        try:
            # stream video frames one by one
            while True:
                stream_bytes += self.rfile.read(1024)
                first = stream_bytes.find(b'\xff\xd8')
                last = stream_bytes.find(b'\xff\xd9')
                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]
                    gray = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
cv2.IMREAD_GRAYSCALE)
                    image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
cv2.IMREAD_COLOR)

                    # lower half of the image
                    height, width = gray.shape
                    roi = gray[int(height/2):height, :]

                    # object detection
                    v_param1 = self.obj_detection.detect(self.stop_cascade,
gray, image)
                    v_param2 = self.obj_detection.detect(self.light_cascade,
gray, image)

                    # distance measurement
```

```
                    if v_param1 > 0 or v_param2 > 0:
                        d1 = self.d_to_camera.calculate(v_param1, self.h1, 300,
image)
                        d2 = self.d_to_camera.calculate(v_param2, self.h2, 100,
image)
                        self.d_stop_sign = d1
                        self.d_light = d2

                    cv2.imshow('image', image)
                    # cv2.imshow('mlp_image', roi)

                    # reshape image
                    image_array = roi.reshape(1, int(height/2) *
width).astype(np.float32)

                    # neural network makes prediction
                    prediction = self.nn.predict(image_array)

                    # stop conditions
                    if sensor_data and int(sensor_data) < d_sensor_thresh: #
                        print("Stop, obstacle in front")
                        self.rc_car.stop()
                        sensor_data = None

                    elif 0 < self.d_stop_sign < d_stop_light_thresh and
stop_sign_active:
                        print("Stop sign ahead")
                        self.rc_car.stop()

                        # stop for 5 seconds
                        if stop_flag is False:
                            self.stop_start = cv2.getTickCount()
                            stop_flag = True
                        self.stop_finish = cv2.getTickCount()

                        self.stop_time = (self.stop_finish - self.stop_start) /
cv2.getTickFrequency()
                        print("Stop time: %.2fs" % self.stop_time)

                        # 5 seconds later, continue driving
                        if self.stop_time > 5:
                            print("Waited for 5 seconds")
                            stop_flag = False
                            stop_sign_active = False

                    elif 0 < self.d_light < d_stop_light_thresh:
```

```
                        # print("Traffic light ahead")
                        if self.obj_detection.red_light:
                            print("Red light")
                            self.rc_car.stop()
                        elif self.obj_detection.green_light:
                            print("Green light")
                            pass
                        elif self.obj_detection.yellow_light:
                            print("Yellow light flashing")
                            pass

                        self.d_light = d_stop_light_thresh
                        self.obj_detection.red_light = False
                        self.obj_detection.green_light = False
                        self.obj_detection.yellow_light = False

                    else:
                        self.rc_car.steer(prediction)
                        self.stop_start = cv2.getTickCount()
                        self.d_stop_sign = d_stop_light_thresh

                        if stop_sign_active is False:
                            self.drive_time_after_stop = (self.stop_start -
self.stop_finish) / cv2.getTickFrequency()
                            if self.drive_time_after_stop > 5:
                                stop_sign_active = True

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        print("car stopped")
                        self.rc_car.stop()
                        break
        finally:
            cv2.destroyAllWindows()
            sys.exit()


class Server(object):
    def __init__(self, host, port1, port2):
        self.host = host
        self.port1 = port1
        self.port2 = port2

    def video_stream(self, host, port):
        s = socketserver.TCPServer((host, port), VideoStreamHandler)
        s.serve_forever()
```

```
    def sensor_stream(self, host, port):
        s = socketserver.TCPServer((host, port), SensorDataHandler)
        s.serve_forever()

    def start(self):
        sensor_thread = threading.Thread(target=self.sensor_stream,
args=(self.host, self.port2))
        sensor_thread.daemon = True
        sensor_thread.start()
        self.video_stream(self.host, self.port1)


if __name__ == '__main__':
    h, p1, p2 = "192.168.1.100", 8000, 8002

    ts = Server(h, p1, p2)
    ts.start()
```

Stream client on raspberry pi code :

```
import
io
import socket
import struct
import time
import picamera


# create socket and bind host
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('192.168.1.100', 8000))
connection = client_socket.makefile('wb')

try:
    with picamera.PiCamera() as camera:
        camera.resolution = (320, 240)      # pi camera resolution
        camera.framerate = 15               # 15 frames/sec
        time.sleep(2)                       # give 2 secs for camera to
initilize
        start = time.time()
        stream = io.BytesIO()
```

```
        # send jpeg format video stream
        for foo in camera.capture_continuous(stream, 'jpeg', use_video_port =
True):
            connection.write(struct.pack('<L', stream.tell()))
            connection.flush()
            stream.seek(0)
            connection.write(stream.read())
            if time.time() - start > 600:
                break
            stream.seek(0)
            stream.truncate()
    connection.write(struct.pack('<L', 0))
finally:
    connection.close()
    client_socket.close()
```

Traffic Signal Code:

```
const int
r = 9;
//connect
red led
at pin 9
            const int y = 10;          //connect yellow led at pin 10
            const int g = 11;          //connect green led at pin 11
            const int sec = 1000;      //seconds defined
            void setup()
              {
                pinMode(r,OUTPUT);
                pinMode(y,OUTPUT);
                pinMode(g,OUTPUT);
                delay(sec);
              }

            void loop()
                {
                    digitalWrite(r,HIGH) ;
                    delay(sec*5);
                    digitalWrite(r,LOW) ;
                    digitalWrite(y,HIGH) ;
                    delay(sec*5);
                    digitalWrite(y,LOW) ;
                    digitalWrite(g,HIGH) ;
                    delay(sec*5);
                    digitalWrite(g,LOW) ;
```

RC control via computer code:

```
//
assign
pin
num
         int right_pin = 6;
         int left_pin = 7;
         int forward_pin = 10;
         int reverse_pin = 9;

         // duration for output
         int time = 50;
         // initial command
         int command = 0;

         void setup() {
           pinMode(right_pin, OUTPUT);
           pinMode(left_pin, OUTPUT);
           pinMode(forward_pin, OUTPUT);
           pinMode(reverse_pin, OUTPUT);
           Serial.begin(115200);
         }

         void loop() {
           //receive command
           if (Serial.available() > 0){
             command = Serial.read();
           }
           else{
             reset();
           }
            send_command(command,time);
         }

         void right(int time){
           digitalWrite(right_pin, LOW);
           delay(time);
         }

         void left(int time){
           digitalWrite(left_pin, LOW);
           delay(time);
         }
```

```
void forward(int time){
  digitalWrite(forward_pin, LOW);
  delay(time);
}

void reverse(int time){
  digitalWrite(reverse_pin, LOW);
  delay(time);
}

void forward_right(int time){
  digitalWrite(forward_pin, LOW);
  digitalWrite(right_pin, LOW);
  delay(time);
}

void reverse_right(int time){
  digitalWrite(reverse_pin, LOW);
  digitalWrite(right_pin, LOW);
  delay(time);
}

void forward_left(int time){
  digitalWrite(forward_pin, LOW);
  digitalWrite(left_pin, LOW);
  delay(time);
}

void reverse_left(int time){
  digitalWrite(reverse_pin, LOW);
  digitalWrite(left_pin, LOW);
  delay(time);
}

void reset(){
  digitalWrite(right_pin, HIGH);
  digitalWrite(left_pin, HIGH);
  digitalWrite(forward_pin, HIGH);
  digitalWrite(reverse_pin, HIGH);
}

void send_command(int command, int time){
  switch (command){

     //reset command
```

```
        case 0: reset(); break;

        // single command
        case 1: forward(time); break;
        case 2: reverse(time); break;
        case 3: right(time); break;
        case 4: left(time); break;

        //combination command
        case 6: forward_right(time); break;
        case 7: forward_left(time); break;
        case 8: reverse_right(time); break;
        case 9: reverse_left(time); break;

        default: Serial.print("Inalid Command\n");
      }
  }
```

Appendix E: Operation Manual

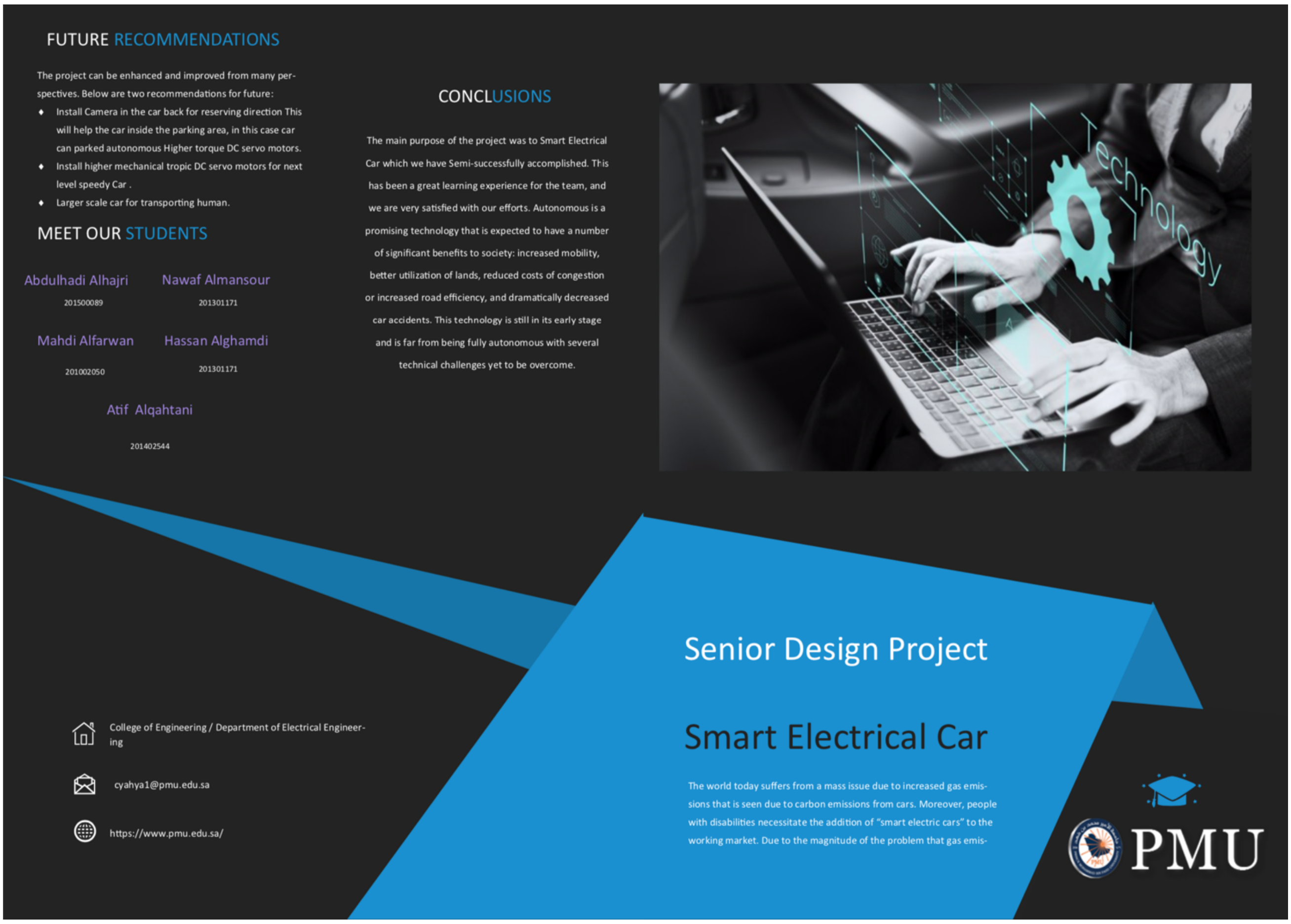
How to drive

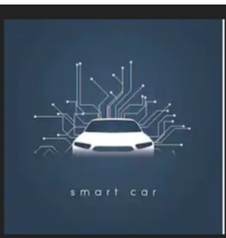
1. **Pi Camera calibration :** Take multiple chess board images using pi camera module at various angles and put them into chess_board folder, run picam_calibration.py and returned parameters from the camera matrix will be used in rc_driver.py.

2. **Collect training/validation data:** First run collect_training_data.py and then run stream_client.py on raspberry pi. Press arrow keys to drive the RC car, press q to exit. Frames are saved only when there is a key press action. Once exit, data will be saved into newly created training_data folder.

3. **Neural network training:** Run model_training.py to train a neural network model. Please feel free to tune the model architecture/parameters to achieve a better result. After training, model will be saved into newly created saved_model folder
4. **Self-driving in action**: First run rc_driver.py to start the server on the computer (for simplified no object detection version, run rc_driver_nn_only.py instead), and then run stream_client.py and ultrasonic_client.py on raspberry pi.

Appendix F: Brochure





PROJECT DEFINITION

To design a smart electric car that will be able to

a. follow a lane in a real road with straight and curved parts,

b. detect obstacles and maneuver around them,

c. recognizes and follows traffic lights

**WE HAVE
GREAT IDEA
FOR SMART CAR**

PROJECT BACKGROUND

With over 110 million vehicle related accidents in the U.S. since 2000, engineers are yet to make a break-through with smart cars. Particularly, there is a requirement to dominate safety mechanisms, automatic cruise control buttons, laser systems and ultimately preventing collisions. Engineers face a huge gap during stabilizing control and assisting by automatically applying breaks. Engineers are yet to make progress in discovering "intelligence transportation" (Helmers & Marx, 2012). Overall, smart cars operate on a laser based sensor that looks at slowing down when another vehicle approaches. Progress is required for intelligent systems to be accepted as mainstream riding options. Researchers suggest that creating and maintaining maps for smart cars is critical work and requires clarification based on autonomous technology.

SYSTEM DESIGN

Safety:

The car should operate safely with acceptable speed without losing the track or hitting objects.

Environmental:

Electric cars have no fuel to burn, therefore no emission of CO 2 and it works on clean environment.

Social:

The system will help people to save the driving time which allows them to be more productive while the car drive itself.

Design Constraints

Safety

Each year 30,000 to 40,000 people killed on the roads in the KSA. Despite a recent decline, there were 247 thousand road accidents in the KSA in 2016 (latest data from the KSA traffic police department/ ministry of interior). Most of these accidents are caused by driver' errors. Driver's mistakes are due to lack of knowledge, failure to comply with the traffic regulations, road distractions, or health issues while driving. All the parts that have been used are considered to be safe to deal with which will be operating with low voltage DC power supply and properly insulated from being exposed which might cause electric shocks or short circuiting.

Environmental

Since the system will reduce the need of human interference, which allows robotics to make smarter decisions which reduce the travelling distance, accidents, and the number of cars needed for individual's transportation. Eventually, this will reduce the CO2 emissions produced by increased travelling distances and the number of cars for each individual. Also, since the number of accidents will decrease, this will enhance the environmental losses due to loss of manufacturing resources.

Social

The system will help people to save the driving time, which allows them to be productive while the car drives. Some studies have been conducted on the influence of robotics over the local society of KSA, which concluded that the society would be able to accept the robotics driven technologies. Yet, there are still some social constrains including convincing the local authorities about the visibility and safety of the

**SAFETY IS
MORE IMPORTANT**

Appendix G: Banner