Smart Autonomous Vehicle in a Scaled Urban Environment

Project Proposal

Team: Devon Bates, Frayne Go, Tom Joyce, Elyse Vernon

Advisor: Dr. José Sánchez

Date: December 10, 2012

Table of Contents

I. Introduction

Autonomous vehicles have become of increasing interest with defense research. The Defense Advanced Research Projects Agency (DARPA) has held its Grand Challenge and Urban Challenge in 2004, 2005, and 2007. These competitions have set some of the most elite schools against one another in a competition to build a fully autonomous vehicle. Recently Google has also announced its "Google Driverless Car"; a fleet of full scale autonomous cars which have logged over 300,000 hours completely accident free [1]. Autonomous vehicles have far reaching benefits; from increasing road safety to allowing those with physical and vision impairments to drive. For the Smart Autonomous Vehicle in a Scaled Urban Environment (SAV-SUE) project, the team will bring the concept of an autonomous vehicle to a small scale.

SAV-SUE will integrate environment detection via a camera input and a digital signal processor (DSP) for image processing with microcontroller-based vehicle control. Figure 1 illustrates a high level system block diagram for the system. The camera will be mounted in the cab in order to mimic a driver's vision. The camera will send pixel data of an image to a DSP to detect the stop sign as well as determine the lane lines. The DSP will interface with the primary controller, which will be a microcontroller. Sensory information will also interface with this primary controller. The primary controller will communicate with the vehicle's Multi-Function Controller (MFC), which will operate the motors and lights to achieve the desired actions of the vehicle. This vehicle will be the Tamiya MAN TGX 26.540 6x4 XLX replica-quality semi truck cab, built to a 1/14-scale.

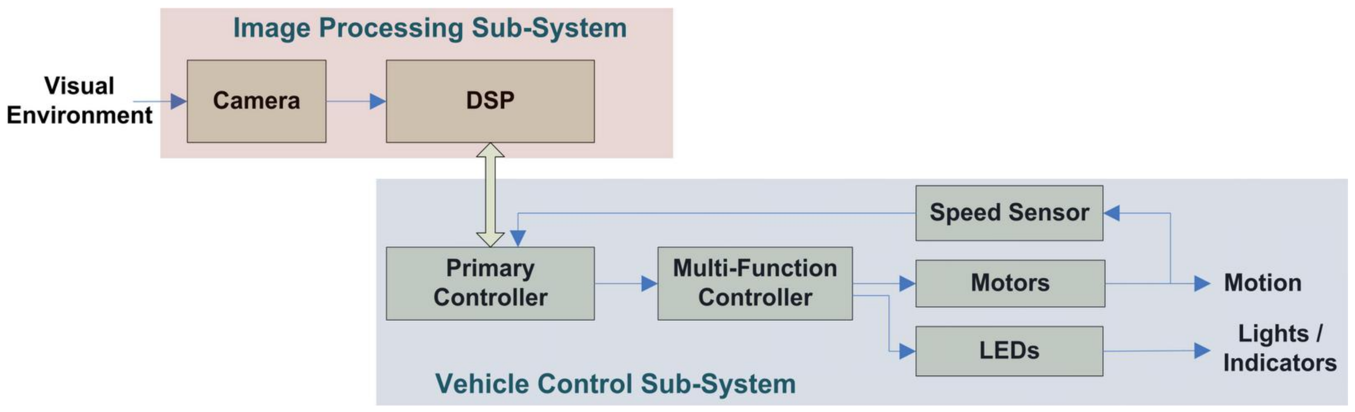


Fig 1: SAV-SUE High Level Block Diagram

II. Objective

Ultimately, the team's objective is to achieve autonomous control such that the vehicle approaches a scaled intersection at cruising speed while staying in the right hand lane, detects a stop sign, stops before a stop line, engages a right turn signal, and executes a right-hand turn into the appropriate lane. This project will be completed to allow for expandability for future senior projects.

III. Functional Description

Though SAV-SUE is primarily focused on operation of its vehicle through vision-based controls, the team must also design the scaled urban environment in which the vehicle is to operate. Based on geometric analysis and vehicle characteristics, the team has determined the size requirements of the scaled roadway intersection. Figure 2 depicts the layout of the environment as it will be constructed in space allotted in Bradley University's engineering building.

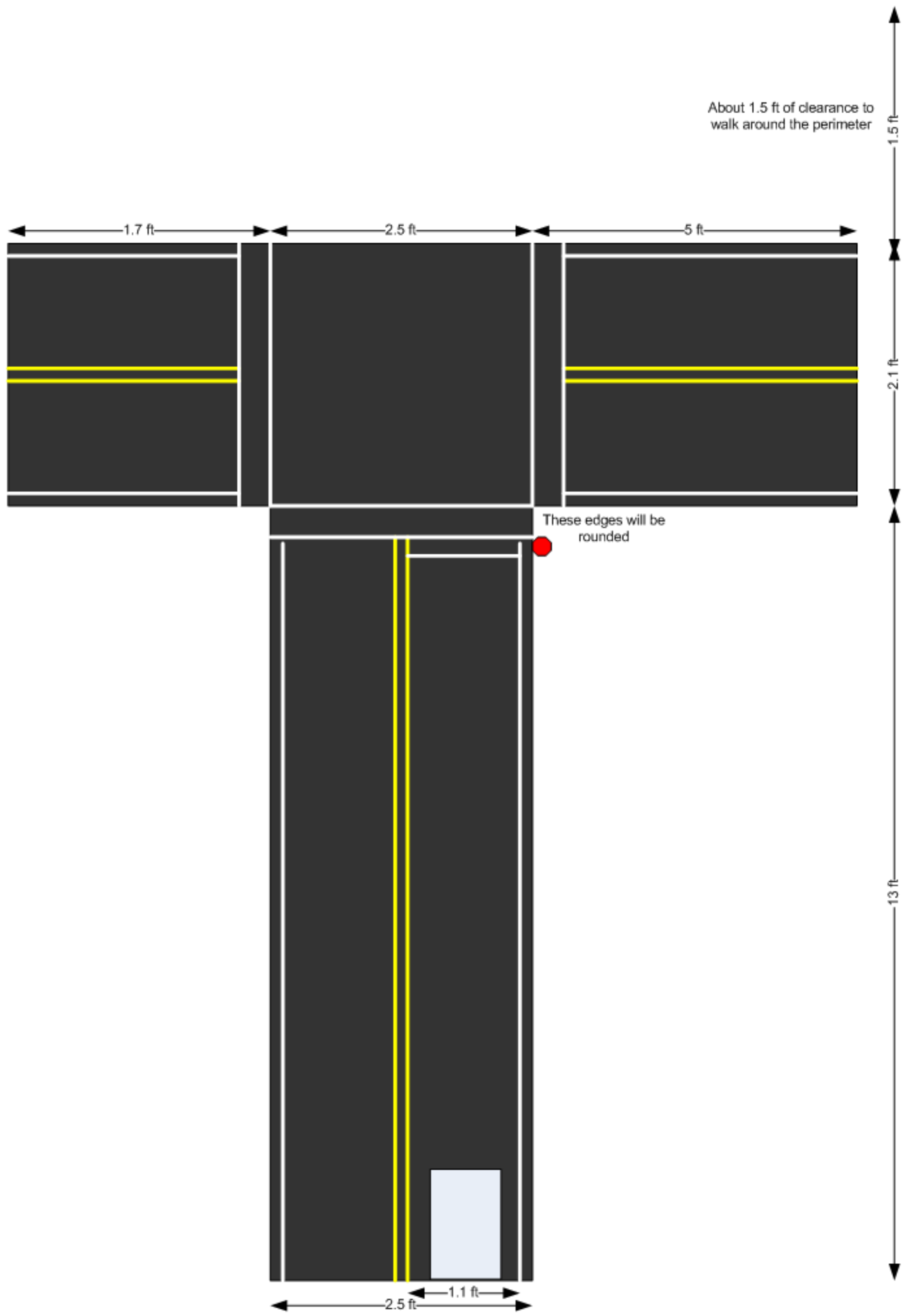

About 1.5 ft of clearance to walk around the perimeter
1.5 ft
1.7 ft
2.5 ft
5 ft
2.1 ft
These edges will be rounded
13 ft
1.1 ft
2.5 ft

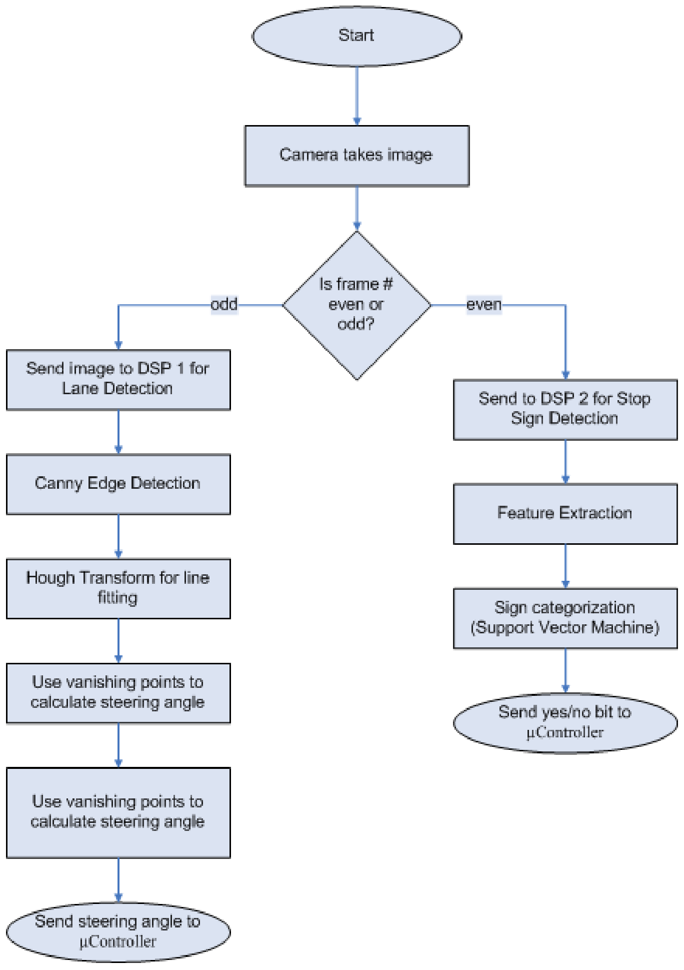
Fig 2: Tentative Scaled Urban Environment Dimensions

A. Image Processing

The main goal for the image processing subsystem is to detect stop signs and traffic lines on the road using image processing algorithms. The image processing calculations will be performed in the DSP unit. The DSP will receive a stream of frames from the camera attached to the car. It will then attempt to recognize a stop sign on the road allowing the control system to respond accordingly. The general form stop sign detection algorithm will take is using a feature descriptor/extractor and a classifier to categorize the contents of the images. The two algorithms the team has investigated for the feature extraction section are Histogram Oriented Gradients (HOG) and ohta space thresholding. The system will use a support vector machine (SVM) to categorize the stop sign feature data regardless of which feature extraction algorithm is used. For the lane line detection the team will use the Canny/Hough estimation of vanishing points (CHEVP) algorithm to calculate the proper steering angle. The team will use MATLAB (Mathworks, Natick, MA) to perform algorithm verification and testing. The algorithm that yields the most accurate and consistent results will be chosen for DSP implementation.

The image processing subsystem must operate in real-time in order t for the control system to react to the environment. To be able to handle the memory needed to store and manipulate images from the camera, the subsystem will use two DSPs connected to the camera. One DSP will handle lane line detection and the other will handle stop sign detection. The camera will alternate sending the image frames to the two DSPs. An important feature is to avoid false negatives, where the system does not recognize a stop sign, in addition to false positives, where the system recognizes a stop sign that does not exist. Possible environmental factors–that the image processing system will have to account for are lighting conditions, stop sign conditions, and image distortions. The algorithm the team utilizes will have to be able to adapt to those conditions and still perform with consistency.

Once the image is processed for stop detection and lane detection, the DSP will trigger an event to the vehicle control subsystem’s primary controller which will be discussed in section three B.



Start
Camera takes image
Is frame # even or odd?
odd
even
Send image to DSP 1 for Lane Detection
Canny Edge Detection
Hough Transform for line fitting
Use vanishing points to calculate steering angle
Use vanishing points to calculate steering angle
Send steering angle to µController
Send to DSP 2 for Stop Sign Detection
Feature Extraction
Sign categorization (Support Vector Machine)
Send yes/no bit to µController

Fig 3: Image Processing High Level Software Flowchart

B. Vehicle Control

The goals of the vehicle control subsystem is to achieve an accurate model of the system, account for that model on the primary controller, reverse engineer the MFC controller to communicate effectively between the primary controller and MFC, and integrate communication data from the DSP and sensory input to achieve autonomous decision making. This subsystem will be responsible for the system outputs, which are vehicle movement and engagement of brakelights and a turn indicator. It will also be responsible for the communication between all system devices including a speed sensor reading from the transmission of the motor.

The vehicle control team will perform extensive modeling and analysis of the system using Simulink (Mathworks, Natick, MA). The modeling will involve determination of motor parameters as well as overall system parameters such as the inertia of the entire truck, which will affect stopping distance. This modelling process will determine parts of the microcontroller software design to achieve the desired system responses.

Motor and turn indicator control will be achieved by reverse engineering an existing, MFC designed by Tamiya Inc. In its intended application, the MFC receives instructions from a radio receiver. The team will use a microcontroller to replicate the de-modulated radio signals that the MFC will recognize. In the beginning stages of the project, the team will reverse engineer the signals to be replicated.

For autonomous control, the primary controller will be communicating with the DSP used for image processing. The DSP will alert the microcontroller about its environment by sending signals upon object detection, and the microcontroller with react accordingly. Specifically, detection of a stop sign will prompt a stopping algorithm stored in code memory. The team will also implement an algorithm used for lane detection to make sure that the vehicle can complete a turn while staying in its lane.

As the microcontroller is operating, it is also receiving sensory feedback for closed-loop control. Ideal closed-loop control serves to eliminate output error by subtracting the difference between desired and actual output performance. An optical sensor on transmission axle shall be used to determine the actual vehicle speed.

IV. Functional Requirements

A. Vehicle Behavior

- The vehicle shall come to a full and complete stop at or before the stop line before proceeding into the intersection.
- The vehicle shall wait for a minimum of 2 seconds before proceeding into the intersection.
- The vehicle shall drive at a speed no more than 3.2 miles per hour.
- While moving down the road the vehicle shall stay within the right hand lane boundaries at all times.
- Upon completion of the right hand turn the vehicle will be positioned entirely in the right hand lane.
- The pressing of a radio-controlled “emergency stop” shall cut power to the entire vehicle.
- The vehicle shall engage a turn signal at a distance of at least 7’ from the stop line.
- The vehicle shall engage its braking lights immediately when the vehicle begins to slow for a stop.
- The priorities of the vehicle behavior will be to stay within the boundaries of the right hand lane and to some to a full and complete stop.

B. Environment

- The scaled urban environment shall require a space no bigger than 10’ x 16’.
- The size of the stop sign shall be no smaller than 2.22” from flat edge to flat edge.

C. Microcontroller

- The main loop of the microcontroller shall be focused on speed control and tracking of the lane.
- The microcontroller shall receive a binary signal from the DSP which shall trigger an interrupt to begin the stopping algorithm.

1. *Communications*

- The microcontroller shall be set up to repeatedly receive from the DSP a variable that describes the distance between the vehicle and the stop line.
- The microcontroller shall be set up to repeatedly receive from the DSP a variable to update the vehicle's steering angle.
- The microcontroller shall be set up to repeatedly receive from the speed sensor a variable that describes the speed of the motor shaft or vehicle.
- The microcontroller shall be set up to repeatedly receive from the DSP a variable that indicates the turning distance at the intersection.

2. *Stop Algorithm*

- The microcontroller shall repeatedly use kinematic equations to calculate the required deceleration to come to a full and complete stop.
- The microcontroller shall output signals to engage the brake lights and turn indicators no closer than 7' from the stop line and shall disengage these signals once it reaches the stop line.
- When the vehicle has come to a full and complete stop the microcontroller shall set a designated bit to notify the microcontroller software that it is time to execute a turn.
- One iteration of the stopping speed update shall execute in 50 ms.
- One iteration to loop through a stopping speed update then to the tracking algorithm shall execute in 200ms.

3. *Turning Algorithm*

- The microcontroller shall instruct the vehicle to proceed forward for a distance specified by the critical distance calculation. This distance shall be met to within 1".
- The microcontroller shall instruct the vehicle to make its optimized 90 degree turn after the vehicle reaches its critical distance.
- Upon completion of the turn the microcontroller shall communicate to the DSP that the turn has be completed and the DSP should resume looking for a stop sign.

4. *Tracking Algorithm*

- The microcontroller shall be able to communicate to the steering servo a desired steering angle given in degrees.
- After adjusting the steering angle of the vehicle the the microcontroller shall wait for no more than 100 ms before adjusting the steering angle back to 0 degrees
- One iteration of the tracking algorithm shall execute in 150 ms.

5. *Speed Control Algorithm*

- The microcontroller shall be able to dictate what gear the motor is operating.

- The speed of the motor shall be increased until the desired speed is obtained.
- The speed of the motor shall be verified and corrected to match a desired value via software.
- One iteration of the speed control algorithm shall execute in 100 ms.
- The vehicle speed shall be updated every 250 ms.

D. Digital Signal Processor (DSP)

- The DSP shall be able to complete the required image processing within 130 ms.
- The DSP shall have USB, UART, I^2C, GPIO and SPI ports.
- The DSP shall fit either stop detection or lane detection algorithm within its flash memory.

1. *Stop Detection Algorithm*

- The DSP shall be able to detect the stop sign from at least 7’ away.
- The DSP shall begin to detect the stop line and calculate the distance to it after detecting a stop sign
- The DSP shall be able to calculate the distance to the stop line from at least 3’ away.
- The DSP shall be able to communicate to the microcontroller a distance variable describing the distance between the vehicle and the stop line.

2. *Lane Detection Algorithm*

- The DSP shall be able to detect the orientation of the vehicle within the lane lines in degrees.
- The DSP shall be able to communicate the orientation of the vehicle to the microcontroller

E. Camera

- The camera shall communicate with two DSPs with I^2C and parallel data connections.
- The camera shall be able to capture a 640 by 480 pixel image..
- The camera shall output data representative of a color image in 8-bit format.
- The camera shall have a transfer rate between 4-8 frames per second.
- The camera shall have a fixed focal length at most 20 mm with an F-stop of 3.5.

V. Preliminary Work

A. Image Processing

1. Stop Sign Detection

The first step in the project’s image processing system is detecting a stop sign. For academic purposes the team propose two methods of stop sign detection algorithms which are ohta space thresholding (OST) with binary gradient and histogram orientation gradient (HOG).

OST is a color segmentation algorithm which extracts certain color blobs from an image. The algorithm achieves its goal by taking the RGB color space and transform it into a two-dimensional orthogonal plane, which a plane is represented by the difference of two uncorrelated colors. This simplifies the calculation between three to two colors. After transforming the color space, thresholding is applied by placing two boundary detection planes. If at a certain pixel location where both orthogonal planes meet or exceed both boundary detection plane, a color element is detected. A binary image thus is created from the detected color elements, with a one indicating an edge has been detected and a zero indicating that no edge has been detected. An binary gradient algorithm was added for detecting the edge of an image.

Binary gradient algorithm is achieved by taking the difference between two adjacent binary pixels. Because the new segmented image consist of binary data, the edge detected image will consist of binary data as well. If the difference between two adjacent pixels at a certain pixel location is 0, there is no edge detected. Vice-versa when the difference is 1, the binary gradient is applied vertically and horizontally for redundancy to ensure that no edges are misidentified.

The histogram of oriented gradients (HOG) is a feature descriptor that is optimized for object detection. The image is described by intensity gradients and edge directions. The algorithm is efficient because there are no color space transformations, all of the calculations are done in the RGB color space. Also, the use of gradients helps the system work invariant to changes in ambient light in the environment. Overall, the HOG technique is a powerful and simple implementation for feature description.

The calculations necessary to perform the HOG algorithm are gradient magnitude and gradient orientation. The magnitude calculation is the difference between adjacent pixels on a given RGB layer. The difference is weighted by the total gradient magnitude over each layer so that the system will remain invariant to ambient light levels. The orientation of the gradient is the direction of the edge, the calculation results in an angle in the range of 0 to 360 degrees, which is then sorted into 8-bins. Based on the system's templates, certain pixels are masked out in each *n x n* sub-block and the rest of the pixels are used to calculate the feature data for the given sub-block. There is a feature data value for each sub-block, bin, and template and these values are used to classify whether or not the image contains a stop sign.

2. *Object Classification*

In order to classify an object by its shape requires the use of support vector machine. A SVM is a classifier that uses non-linear mapping of object instances. A linear hyperplane that contains the maximal margins between classes is then used to classify the data. The team has simulated a SVM in MATLAB using the results of the HOG feature description. The SVM trains the system by finding the maximal margins based on the given training set and then uses the results of the training to test subsequent images by comparing them to pre-determined hyperplane. The current implementation of the SVM uses a 30 image training set and has a success rate of 80% on a test set of 15 images.

3. Lane Line Detection

The SAV-SUE team propose implementing what is known as the B-snake technique. The first step is using Canny/Hough estimation of vanishing points (CHEVP). The Canny edge detection locates edges in the image and results in a binary image. Next, the Hough transform uses the binary edge data to determine the equations for any lines present in the image. The line calculations can be used to determine the vanishing point which can then be used to calculate the orientation of the vehicle within the lane lines. The canny edge detection has been simulated in MATLAB successfully, but the Hough transform has yet to be finished.

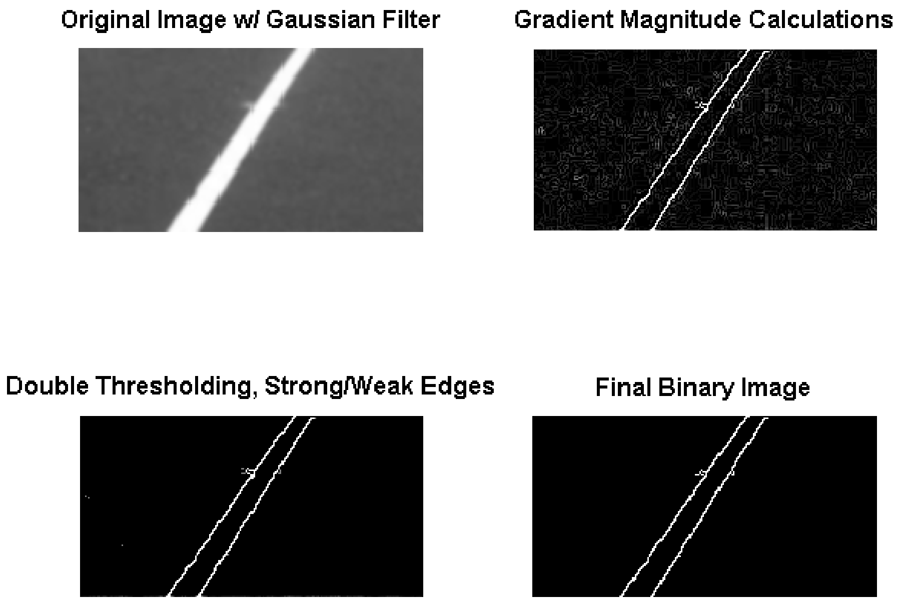

Original Image w/ Gaussian Filter
Gradient Magnitude Calculations
Double Thresholding, Strong/Weak Edges
Final Binary Image

Fig. 4 Results of Canny Edge Detection

B. Vehicle Behavior

1. Environment Design

In order to have an environment in which the vehicle can operate, the team will have to build a miniature roadway. The objective of environment design is to ensure that the truck will be able to turn in the 10’x16’ space allotted in for the project in senior lab. To make sure that the team could meet this requirement, the vehicle’s turning radius was calculated. Figure 5 below shows the equation used to calculate this turning radius. This calculation involved knowing the dimensions of the wheelbase of the truck *WB* as well as the width of the truck *W*. The team also had to know the steering angle of the truck, though an estimation was used. In order to chose the worst case estimation, it was assumed that the steering angle of the truck was 30 degrees. Based on videos from the product's website, this was the smallest turn angle of the truck. It was imperative to choose the smallest predicted turn angle because a smaller turn angle yields a larger turn radius, thus producing a worst case scenario.

$$R = \frac{WB}{\sin(\theta)} + \frac{W}{2}$$

Fig 5: Equation used for the turn radius calculation

In Fig. 6 the diagram illustrating the results of the turn radius calculation is shown. The value for "L" is the value for the turn radius which is equal to 2'6.5". This value is more than enough for the vehicle to be able to turn in the space allotted.

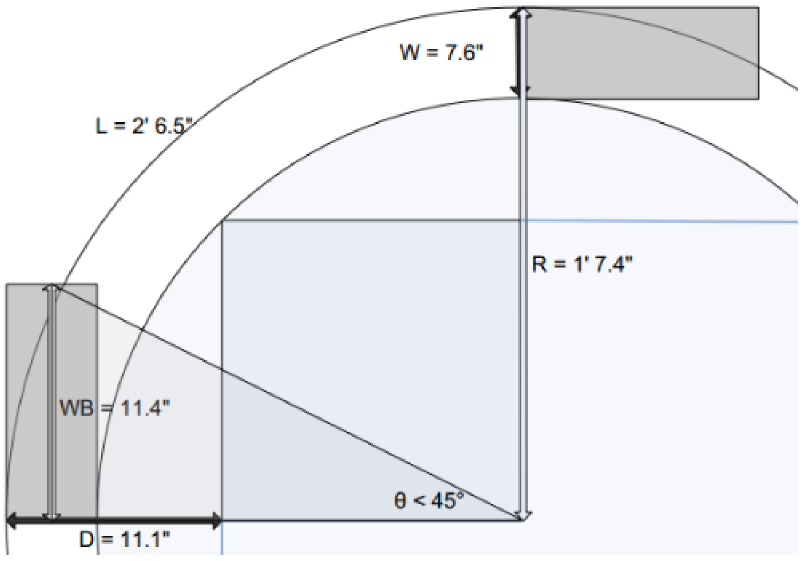


Fig 6: Design of the turn radius

2. Motor Modeling

To be able to control the speed of the truck, the team must model the motor. To begin this process, the team started by experimentally determining the Mabuchi 540 motor parameters. The most critical parameter was the armature resistance, R_a because all of the other parameters relied on this value. In order to calculate this value, the shaft of the motor was stopped and the input voltage and current were measured. This measurement was repeated multiple time in order to increase the accuracy of the measurement. Once the resistance was calculated accurately, the time constant of the step response was used to determine the armature inductance. All of the final values for the motor parameters is shown in Figure 8.

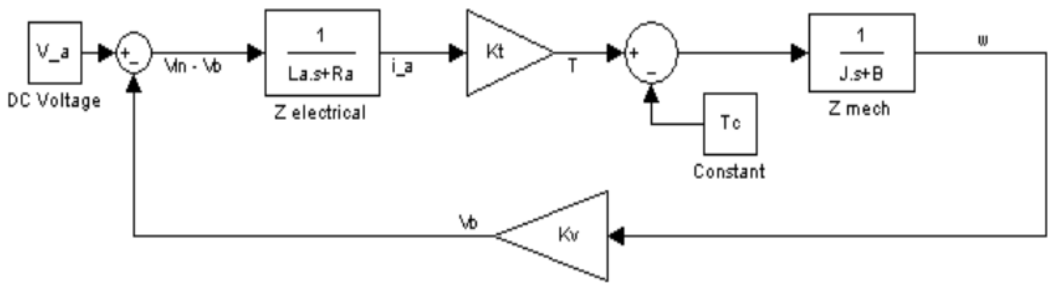


Fig. 7: Generic Motor Modeling Block Diagram

R_a = 1.91E-1	L_a = 3.25E-5	$K_v = K_t$ = 3.90E-3
B = 9.12E-7	T_c = 2.87E-3	J = 6.48E-8

Fig. 8: Experimentally Determined Motor Parameters

Once the motor parameters were determined experimentally, they were entered into the generic control block diagram for the motor shown in Fig. 7. The comparison of this control block diagram model and the actual speed of the motor is shown in Fig. 9. The green bubble points indicate the actual speed of the motor and the dashed black line indicates the Simulink model. The lower voltage range deviates significantly between simulation and measurements; but for the system's operating voltage between 6 and 7 volts, the model is within 1.5% of experimental measurements.

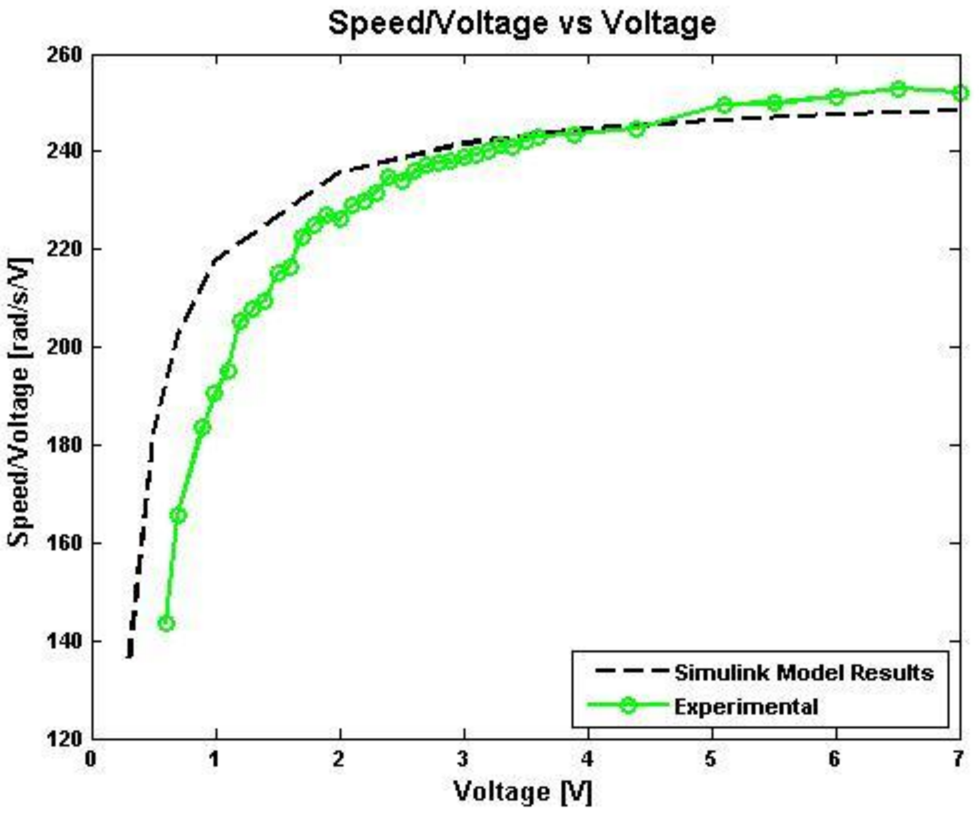


Fig. 9: Graph comparing the actual results from the motor and out Simulink model

The motor model in Fig. 9 only considers the motor without the transmission attached. The transmission includes three different gears that will vary the speed of the vehicle's axle. Therefore, the model must be appended to include these gear transforms.

3. Software Design

The high level flowchart for the primary controller software is shown in Fig. 9. The main loop for the microcontroller software will execute a speed control and tracking algorithm. Another important feature of the microcontroller software is the "Stop Detection Interrupt." This interrupt will be triggered by the a signal received from the digital signal processor which will indicate to the the microcontroller that a stop has been detected and begin executing a stop algorithm to slow down the vehicle. The tracking algorithm also needs to be included in the stop detection interrupt in order to make sure the truck stays within the right hand lane. In order to know when it is time to make the turn there is a turn bit within the stopping algorithm that will be set once the vehicle has come to a full and complete stop at the stop line. As a safety feature, a radio signal will be configured to cut power to the system to protect the equipment and its operators.

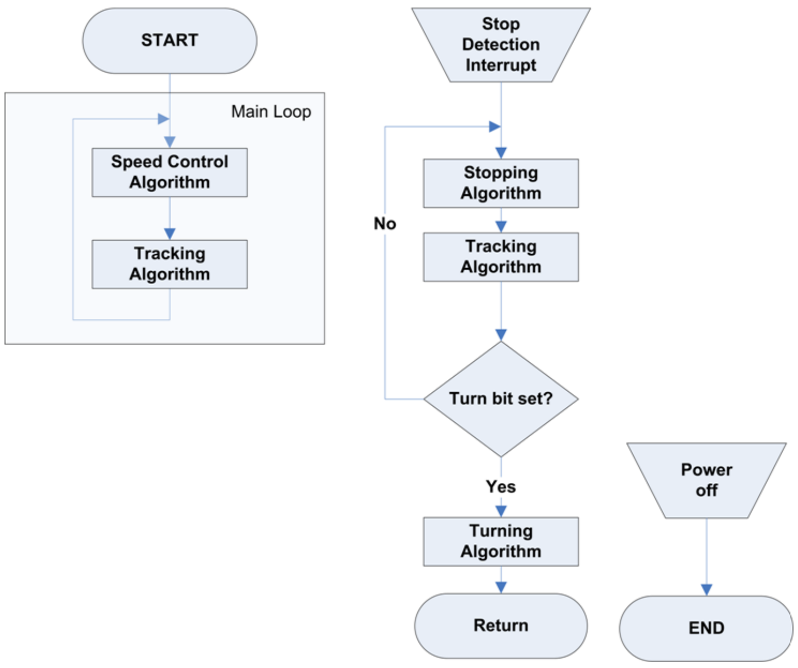


Fig. 10: High level microcontroller flowchart

i. Speed Control Algorithm

The speed control algorithm will use the a speed sensor mounted to the transmission output to compare desired speed with actual vehicle speed. It also relies on an accurate model of the three transmission gear speeds. Note that an iteration of the algorithm must execute in less than 100 ms and that the tracking algorithm must execute in less than 150 ms, making the main loop iterate at 250 ms or 4 Hz. The software flowchart for the speed control algorithm is shown in Figure 10.

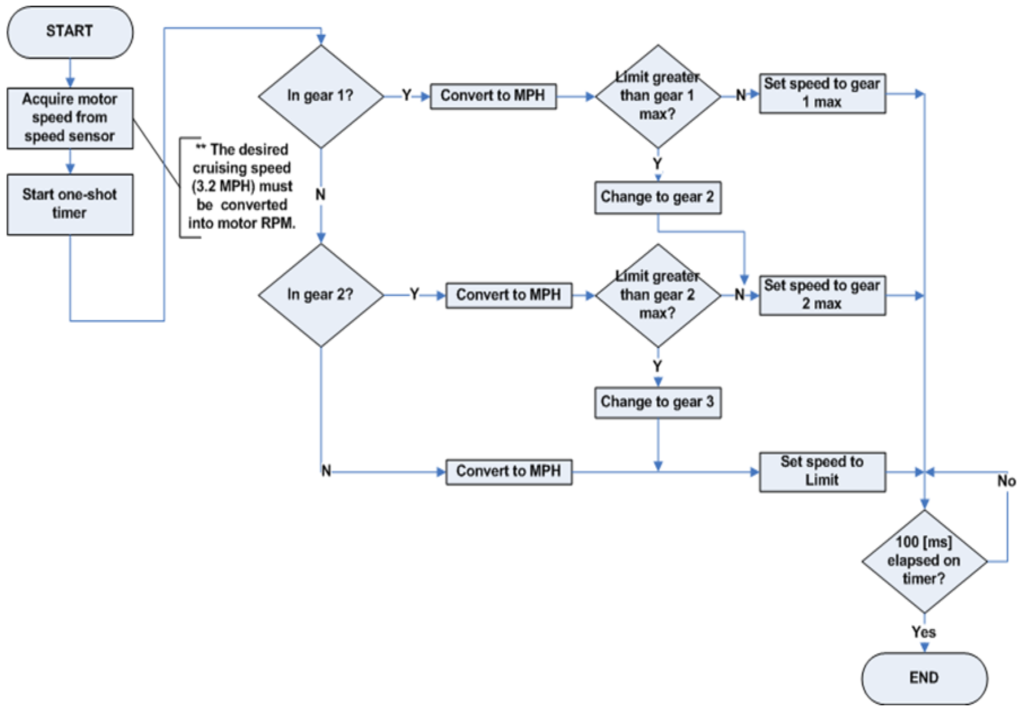


Fig. 11: Speed control algorithm

ii. Tracking Algorithm

The system's tracking algorithm assumes that a corrective steering angle can be calculated by the digital signal processor and sent to the microcontroller across the communication bus. The corrective angle shall be sent to the steering servo for a given amount of time before the steering servo returns to a zero degree steering angle as indicated in the steering angle coordinate system in Fig. 11. The time before the servo rights to a zero degree steering angle will be a function of the magnitude of the steering angle and inversely proportional to the inertia and responsiveness of the vehicle. A definitive number will be determined once the frame of the truck has been built.

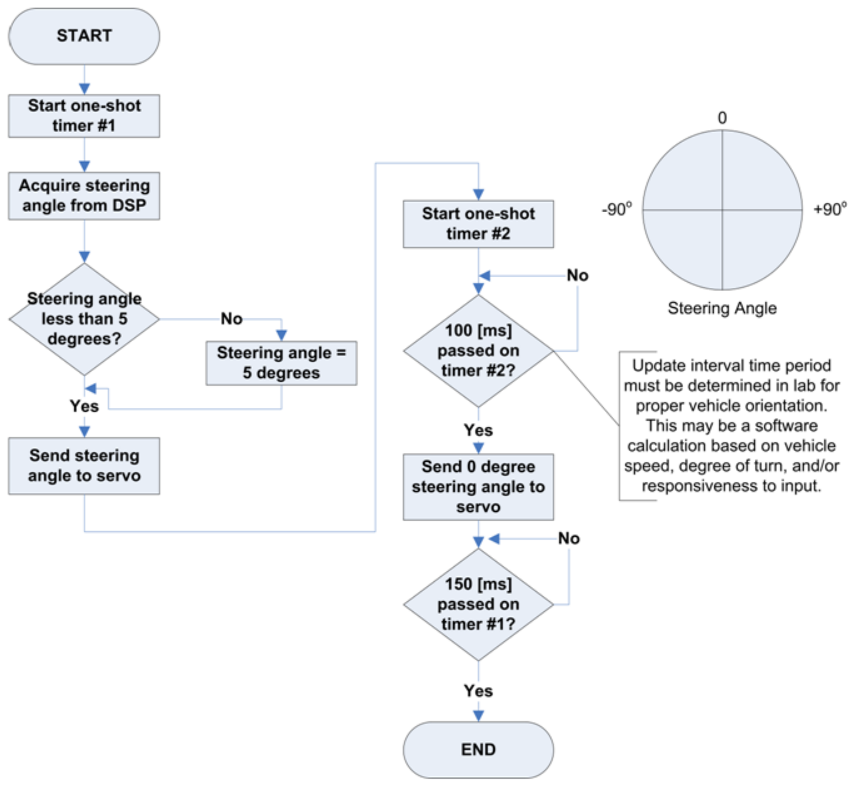

START
Start one-shot timer #1
Acquire steering angle from DSP
Steering angle less than 5 degrees?
No
Steering angle = 5 degrees
Yes
Send steering angle to servo
Start one-shot timer #2
No
100 [ms] passed on timer #2?
Yes
Send 0 degree steering angle to servo
No
150 [ms] passed on timer #1?
Yes
END
0
-90°
+90°
Steering Angle
Update interval time period must be determined in lab for proper vehicle orientation. This may be a software calculation based on vehicle speed, degree of turn, and/or responsiveness to input.

Fig. 12: Tracking Algorithm

iii. Stopping Algorithm

Once a stop sign has been detected and the program counter is within the stopping loop, the program shall execute the stopping algorithm in less than 50 ms and the tracking algorithm in less than 150 ms iteratively for a total loop time of 200 ms. The stopping algorithm assumes the reception of a distance variable from the digital signal processor that describes the distance between the vehicle and the stop line. It shall also use a speed sensor sample to determine the required deceleration rate of the vehicle such that the vehicle comes to a stop at the stop line. The loop exit condition is for the distance between the vehicle and stop line to be zero. At this point, the software shall check that the vehicle speed is zero and apply immediate braking as necessary before setting a turn bit to begin the turning algorithm.

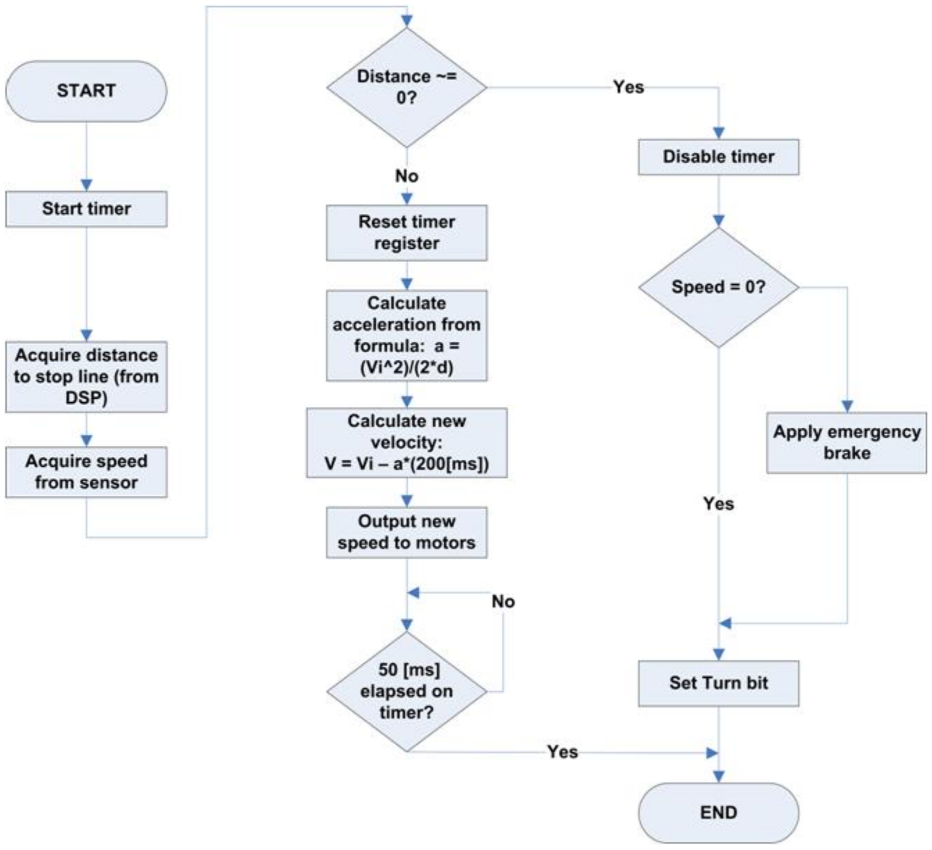

START
Start timer
Acquire distance to stop line (from DSP)
Acquire speed from sensor
Distance ~= 0?
Yes
No
Reset timer register
Calculate acceleration from formula: a = (Vi^2)/(2*d)
Calculate new velocity: V = Vi – a*(200[ms])
Output new speed to motors
No
50 [ms] elapsed on timer?
Yes
Disable timer
Speed = 0?
Apply emergency brake
Yes
Set Turn bit
END

Fig. 13: Stopping algorithm

iv. Turning Algorithm

The turning algorithm shall rely on prior knowledge of the intersection's turn radius. The critical turn distance of the vehicle shall be determined based on the vehicle's turn radius and the distance from the front of the truck to the rear axle, shown in Fig. 13 as the "Optimized 90 degree turn." The critical turn distance will be the distance that the vehicle shall proceed in a straight direction before engaging the steering servo. Once software determines that the vehicle has moved for this appropriate distance, the steering servo shall be engaged to its maximum angle until the vehicle has made a 90 degree turn. Software shall then return the program counter to the main loop such that speed control and tracking resume.

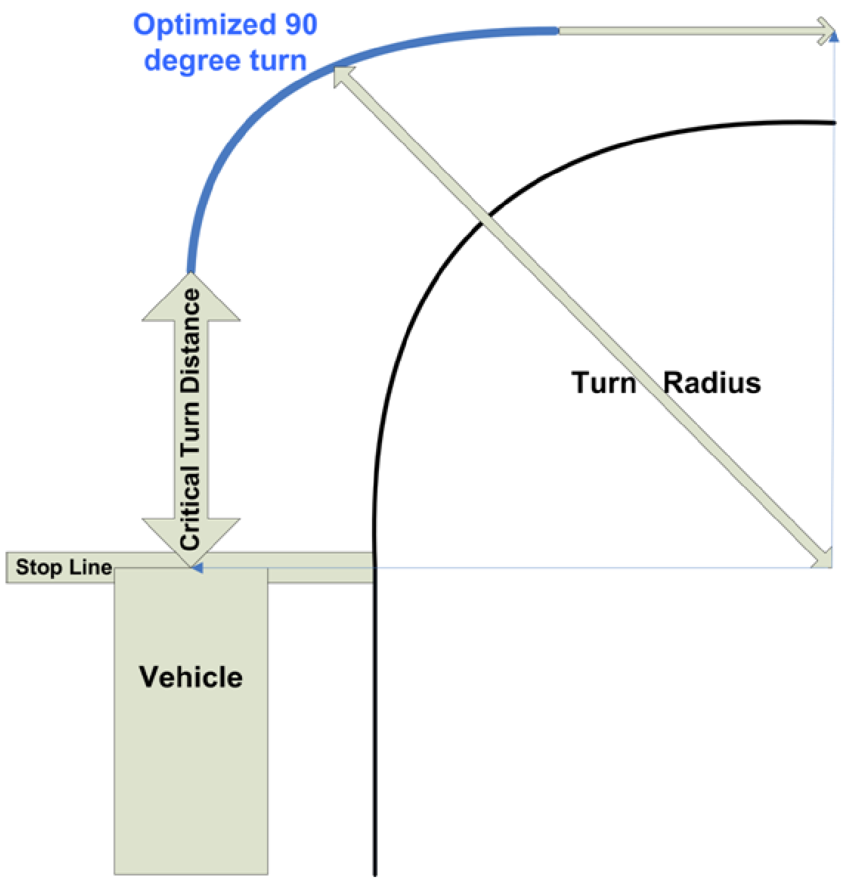
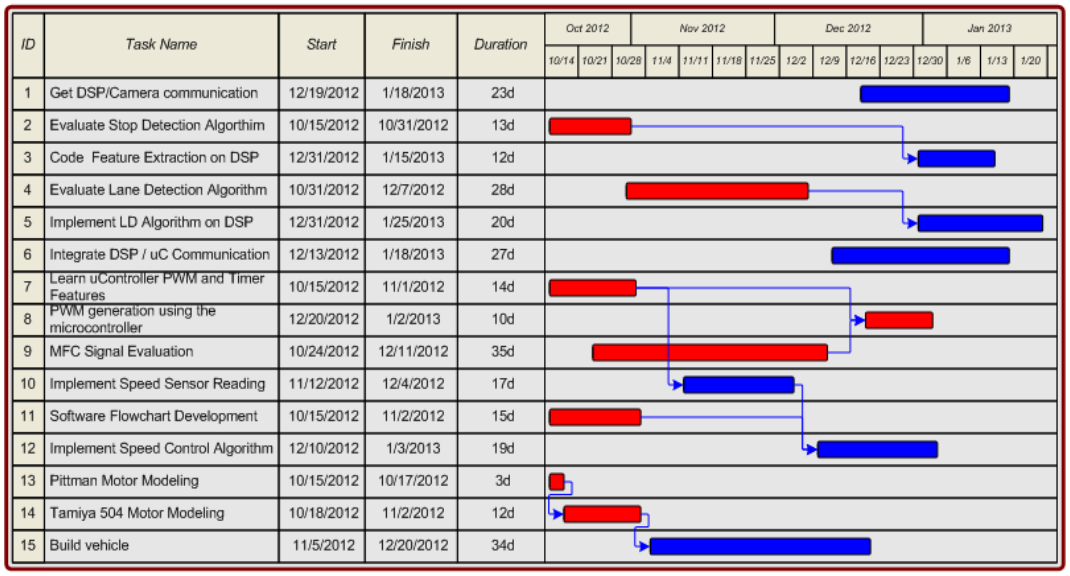


Fig. 14: Turning Algorithm Concept

VI. Tentative Schedule

A. Fall 2012

Figure 15 shows the schedule for Fall Semester 2012. The items colored red have been completed: evaluating the stop sign detection algorithm, evaluating the lane detection algorithm, learning the microcontroller PWM and timer features, generating a PWM signal using the microcontroller, evaluating the MFC signals, software flowchart development, and motor modeling for the Pittman and 540 motor. The following tasks are partially complete: building the vehicle, implementing the sensor reading, coding the speed control algorithm, coding the stop sign detection algorithm, and coding the lane detection algorithm. Based on completed and partially completed progress, the project is somewhat behind schedule and will have to push some items to the Spring Semester 2013.

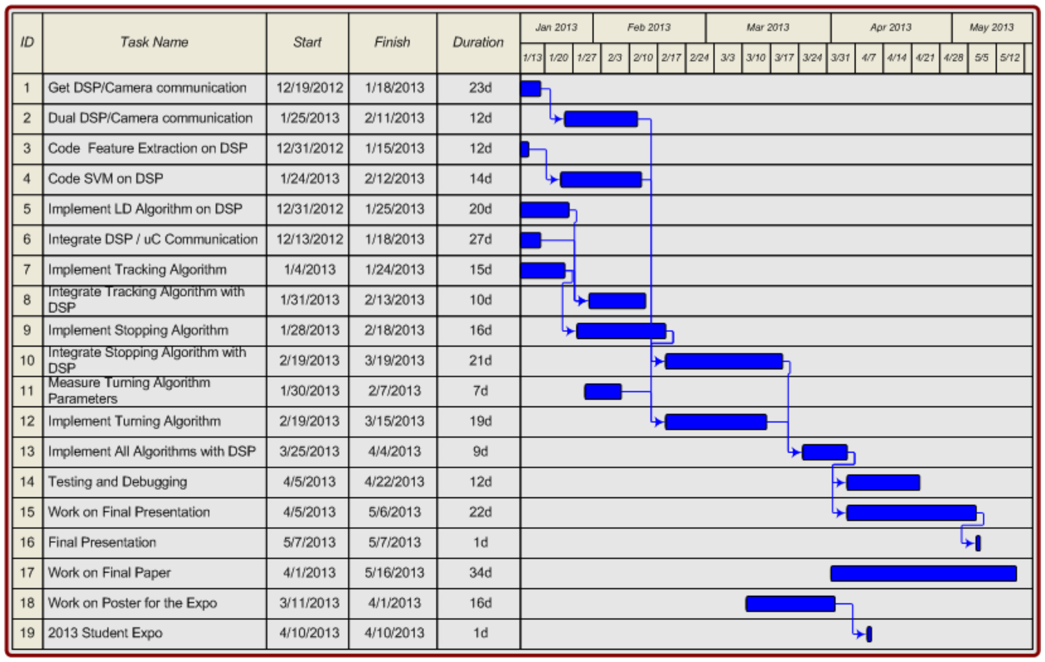


ID	Task Name	Start	Finish	Duration
1	Get DSP/Camera communication	12/19/2012	1/18/2013	23d
2	Evaluate Stop Detection Algorthim	10/15/2012	10/31/2012	13d
3	Code Feature Extraction on DSP	12/31/2012	1/15/2013	12d
4	Evaluate Lane Detection Algorithm	10/31/2012	12/7/2012	28d
5	Implement LD Algorithm on DSP	12/31/2012	1/25/2013	20d
6	Integrate DSP / uC Communication	12/13/2012	1/18/2013	27d
7	Learn uController PWM and Timer Features	10/15/2012	11/1/2012	14d
8	PWM generation using the microcontroller	12/20/2012	1/2/2013	10d
9	MFC Signal Evaluation	10/24/2012	12/11/2012	35d
10	Implement Speed Sensor Reading	11/12/2012	12/4/2012	17d
11	Software Flowchart Development	10/15/2012	11/2/2012	15d
12	Implement Speed Control Algorithm	12/10/2012	1/3/2013	19d
13	Pittman Motor Modeling	10/15/2012	10/17/2012	3d
14	Tamiya 504 Motor Modeling	10/18/2012	11/2/2012	12d
15	Build vehicle	11/5/2012	12/20/2012	34d

Fig. 15: Fall 2012 Proposed Schedule

B. *Spring 2013*

Figure 16 illustrates the Spring Semester 2013 schedule. Some the items from the Fall schedule that may have to be pushed into the Spring schedule include the following: coding the lane detection algorithm, coding the stop sign detection algorithm, and coding the speed control algorithm. Pushing these items from the Fall schedule to the Spring schedule will likely result in a difficult Spring schedule since most of the items in the schedule are linked.



ID	Task Name	Start	Finish	Duration
1	Get DSP/Camera communication	12/19/2012	1/18/2013	23d
2	Dual DSP/Camera communication	1/25/2013	2/11/2013	12d
3	Code Feature Extraction on DSP	12/31/2012	1/15/2013	12d
4	Code SVM on DSP	1/24/2013	2/12/2013	14d
5	Implement LD Algorithm on DSP	12/31/2012	1/25/2013	20d
6	Integrate DSP / uC Communication	12/13/2012	1/18/2013	27d
7	Implement Tracking Algorithm	1/4/2013	1/24/2013	15d
8	Integrate Tracking Algorithm with DSP	1/31/2013	2/13/2013	10d
9	Implement Stopping Algorithm	1/28/2013	2/18/2013	16d
10	Integrate Stopping Algorithm with DSP	2/19/2013	3/19/2013	21d
11	Measure Turning Algorithm Parameters	1/30/2013	2/7/2013	7d
12	Implement Turning Algorithm	2/19/2013	3/15/2013	19d
13	Implement All Algorithms with DSP	3/25/2013	4/4/2013	9d
14	Testing and Debugging	4/5/2013	4/22/2013	12d
15	Work on Final Presentation	4/5/2013	5/6/2013	22d
16	Final Presentation	5/7/2013	5/7/2013	1d
17	Work on Final Paper	4/1/2013	5/16/2013	34d
18	Work on Poster for the Expo	3/11/2013	4/1/2013	16d
19	2013 Student Expo	4/10/2013	4/10/2013	1d

Fig. 16: Spring 2013 Proposed Schedule

VII. Equipment List

- Tamiya RC MAN TGX 26.540 6x4 XLX
- Tamiya RC Multi Function Control Unit (MFC)
- TMS320C5515 eZdsp
- Code Composer Studio
- I^2C VGA Camera
- Mabuchi 540 Motor
- Stellaris Launch Pad Microcontroller
- MATLAB/Simulink
- Wood & Paint to build the Environment

VIII. References

[1] Owen Thomas. (2010 September 7). "Google's Self-Driving Cars May Cost More Than A Ferrari." [Online]. Available: <http://www.businessinsider.com/ google-self-driving-car-sensor-cost-2012-9>.

[2] H. Gomez-Moreno et. al. “Goal Evaluation of Segmentation algorithms for Traffic Sign Sign Recognition,” Intelligent Transportation Systems, IEEE Transactions, 2010, vol. 11, no. 4, pp.917-930.

[3] J. C. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines”. Microsoft, Tech. Rep. MSR-TR-98-14, 1998.

[4] J. Kroll. (2010, September 24). “How to Calculate a Turning Circle” [Online]. Available: <http://www.ehow.com/how_72257 84_calculate-turning-circle.html>.

[5] J. White. (2011 December). "2012 Illinois Rules of the Road" [Online]. Availavle: <http://www.cyberdriveillinois.com/publications/pdf_publications/dsd_a112.pdf>.

[6] K. Curtis. (2012 August 19). "Using Requirements Planning in Embedded Systems Design.” [Online]. Available: <http://new.eetimes.com/design/industrial-control/4206344/Tutorial-on-using -requirements-planning-in-your-embedded-systems-design-Part-1>.