Autonomous Vehicle Control System

Final Design
(Deliverable #8)

GROUP #1:

Colin Lobo
Mohit Bhagotra
Steven Back
Gregory Mainse
Hector Valdez
Matthew Dawson

March 10, 2012
Revision 3.0

Revision History

Revision Number	Author(s)	Description	Date (D/M/Y)
0.1	Steven Back	Created main template for document	04/11/11
0.2	Steven Back	Added individual components (Arduino side)	07/11/11
0.3	Mohit Bhagotra	Added individual components (BB-xM side)	10/11/11
0.4	Colin Lobo	Added individual components (Serial Comm.)	11/11/11
0.5	Matt Dawson	Added individual components (BB-xM side)	17/11/11
0.6	Greg Mainse	Added individual components (BB-xM side)	17/11/11
0.7	Hector Valdez	Added individual components (Arduino side)	17/11/11
0.8	Mohit, Steven, Colin	Added content for various sections	18/11/11
0.9	Mohit Bhagotra	Refined System Context Diagram	19/11/11
0.10	Colin Lobo	Added/Update content to various sections	19/11/11
0.11	Steven Back	Refined individual components (sensors)	19/11/11
1.0	Colin Lobo	Formatted document and final edit for DSD (Deliverable #2) submission	20/11/11
1.1	Colin Lobo	Appended MIS for each component completed by individual team members and Data Flow Diagram	07/01/12
1.2	Steven Back	Fixed some naming discrepancies between MIS modules, and fixed some spelling mistakes	08/01/12
1.3	Greg Mainse	Revised document for spelling and grammar, fixed naming and formatting inconsistencies in MIS	08/01/12
2.0	Steven Back	Made the component design document, fixed formatting	20/01/12
2.1	Steven Back	Added Matt's content, references, formatting	21/01/12
2.2	Colin Lobo	Added up-to-date information regarding sensor considerations for sections: encoder, rear range finder, front range finder. Added new section 4.4.1 Camera considerations.	22/01/12
2.3	Mohit Bhagotra	Updated transmission module to clarify packet size and purpose of spare bits..	22/01/12

2.4	Steven Back	Beagleboard Timing information added, fixed formatting throughout	22/01/12
2.5	Steven Back	Changed document title to final design deliverable. Updated Encoder, Ultra-Sonic. Removed Obstacle Detection and speed_Encoder.	10/03/2012
2.6	Mohit Bhagotra	Updated the SerialComm (A->B) and SteeringControl modules to reflect current interface and implementation.	11/03/2012
3.0	Colin Lobo	Updated sensor placement and sensor considerations sections. Submitted document for Deliverable #8 (System Design Final Rev)	11/03/2012

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Document Purpose

The purpose of this document is to effectively communicate the overall design of a system which meets the requirements stated in the SRS. This includes the development team's decisions and considerations for the system's components. The reader should by the end of this document have a clear understanding of:

1. How the system interacts with its environment
2. How the system is broken down into its components and their roles
4. Timing constraints that are present with the system design
5. Build issues that are possible due to the choice of system design
6. What is defined as normal operation and undesired events likely to occur

1.2 System Scope

1.2.1 Purpose

The purpose of the project is to provide the client a well-documented and complete engineering system that completes the following tasks:

1. A system that will operate autonomously
2. A system that will follow and stay within a lane of a predefined track (REF SRS 2.4) at all times
3. A system that will avoid inanimate obstacles in its path
4. The system's final design will cost no more than $750.00

1.2.2 Goals

Found below are the goals for this project in no particular order. Further explanations of each individual goal can be found in the SRS (REF SRS 1.2.2).

	GOAL (not in any particular order)
1	KISS – Keep it Simple Stupid
2	Don't crash
3	Stay within the boundaries of road
4	Stay completely within one lane when not changing lane
5	Drive forward when not avoiding obstacles
6	Emergency stop
7	Go as fast as possible without issues
8	Smooth motion of vehicle
9	Advertisement opportunities- money donated if profit
10	Follow other moving object

11	Look good
12	Stay within budget
13	Cost effective
14	Good code
15	Indicate lane change

Table 1: List of goals for project

1.2.3 Project Scope

The scope of this project is to test the ability of students in the Mechatronics and Software Engineering disciples to culminate all that they have learned in their academic careers and apply it to a major year-long project. The development team will be designing a system (1/10 scale car) which can autonomously navigate and avoid obstacles in a closed environment provided by the client. Any functionality other than described in the SRS is considered out of the project scope.

1.3 Intended Audience and Document Overview

The intended client of this document shall be Dr. Alan Wassyng, while the audience also comprises of Dr. Wassyng's teaching assistants and the project developers. The document is thus tailored towards a technical, well versed, audience that is familiar with basic technical terminology. The document is also tailored towards the developers of the project and is thus organized in a very methodical structure to allow for the design and final product to reference the DSD with ease and create a seamless transition between documents.

1.4 Document Conventions

1.4.1 Naming Conventions

The following naming conventions are observed in this document:

k_: constant value
m_: monitored variable
c_: controlled variable
e_: enumerated values
y_: enumeration
i_: input variable (individual component)
o_: output variable (individual component)
d_: data variable (data in communications packet)
s_: data structures
t_: data types

The first letter of the constant shall be lower case, and all subsequent starting characters are upper case:

Ex. k_MyDogSkip

Previous values shall be represented by a subscript "-x" where x represents how far in the past

Ex. k_MyDogSkip$_{-2}$

1.4.2 Formatting Conventions

- **Paper size:** US letter (8.5"x11")
- **Margins:** top margin: 0.6", bottom margin: 0.5 ", inner margin: 0.75", outer margin: 0.75", header and footer 0.3" from edge.
- **Header:** Each page shall have a header with the following attributes:
  - Font and size Times (New) Roman, Bold, 14 point for portrait oriented documents
  - Font and size Times (New) Roman, Bold, 18 point for landscape oriented documents (e.g. PowerPoint)
  - Line below, with 2 points separation from text
  - Left, aligned with margin: the month and year of the publication (the venue date)
  - Right aligned to the margin: the document designator, which includes the document number:
  - doc.: AVCS_XX_RevY
  - Where:
    - XX is the abbreviation for the document (SRS – System Requirements Specification)
    - Y is the revision number
- **Footer:** Each page shall have a footer with the following attributes:
  - Font and size Times (New) Roman, Normal, 12 point
  - Line above
  - Left, aligned with margin: the word "Submission"
  - Center: the word "page" followed by the page number
  - Right aligned to the margin: the first author and company (in the format: author_name, company).
- Every document submission must have an author and company as a point of reference for the submission.

1.5 References and Acknowledgements

- The formatting guidelines have been adapted from the IEEE 802.22 Documents guideline

Example: Internet URL:

Author(s)*. “Title.” Internet: complete URL, date updated* [date accessed].
M. Duncan. “Engineering Concepts on Ice. Internet: www.iceengg.edu/staff.html, Oct. 25, 2000 [Nov. 29, 2003].

2. OVERALL DESIGN DESCRIPTION

2.1 System Functionality

Two main functionalities that encapsulate the entire autonomous system are as follows:

1. The system will have lane following ability.
2. The system will avoid obstacles.

The following is a system design on implementing this functionality.

2.2 System Context

The below diagram depicts a high level design of the kind of monitored variables entering the system and the control variables leaving the system. We are monitoring the position of the obstacles and lanes in reference to the system, as well as the system’s speed in relation to the environment. Likewise, we are controlling the speed of the motor and the stop control through the speed controller and the steering control of the system through the steering servo.

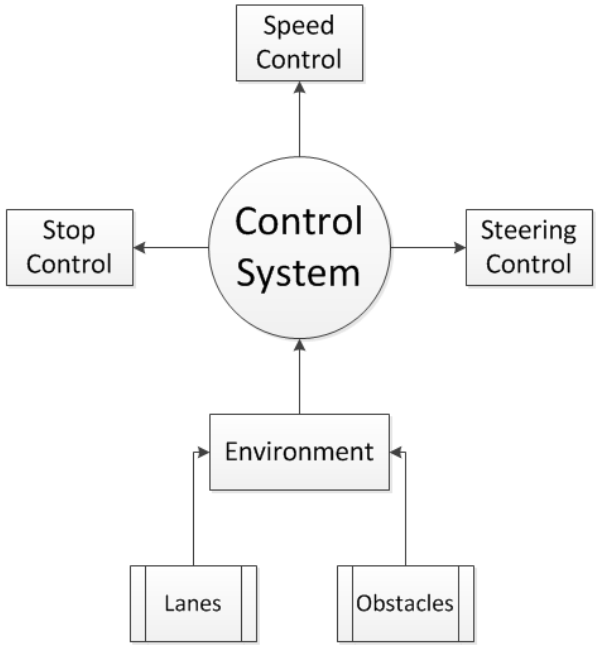


Figure 1: System Context Diagram (REF SRS 2.8)

2.3 Overview of System

The diagram below shows an overview of the system level design. The red box represents the system and everything outside represents the environment. The system is broken down into several high level components. These include: vehicle chassis, speed controller, steering controller, various sensors, BeagleBoard-xM, and an Arduino Mega.

The vehicle chassis will simply contain all other components and will be the component which navigates the track. The speed controller will be the main component controlling the vehicles speed (Figure 1 Speed Controller). The steering controller will be the main component controlling the vehicle's steering (Figure 1 Steering Control).

The sensor components represent the systems sensors that monitor the environment. These include but are not limited to encoders, obstacle detection sensors, range finders, and a camera. The Arduino Mega represents the controller. Its purpose is broken down into two main roles. The first role is to accept in monitored variables, convert them into input variables that represent real world factors, and package the input variables to be sent along serial. The second role is to accept serial packages, unpack them, convert the output variables to control variables and send them to either the steering servo or the speed controller.

The last component of the system is the BeagleBoard-xM, the central processor. It completes three main roles; camera communication, serial communication, path choice and corresponding control variable outputs to complete that path choice.

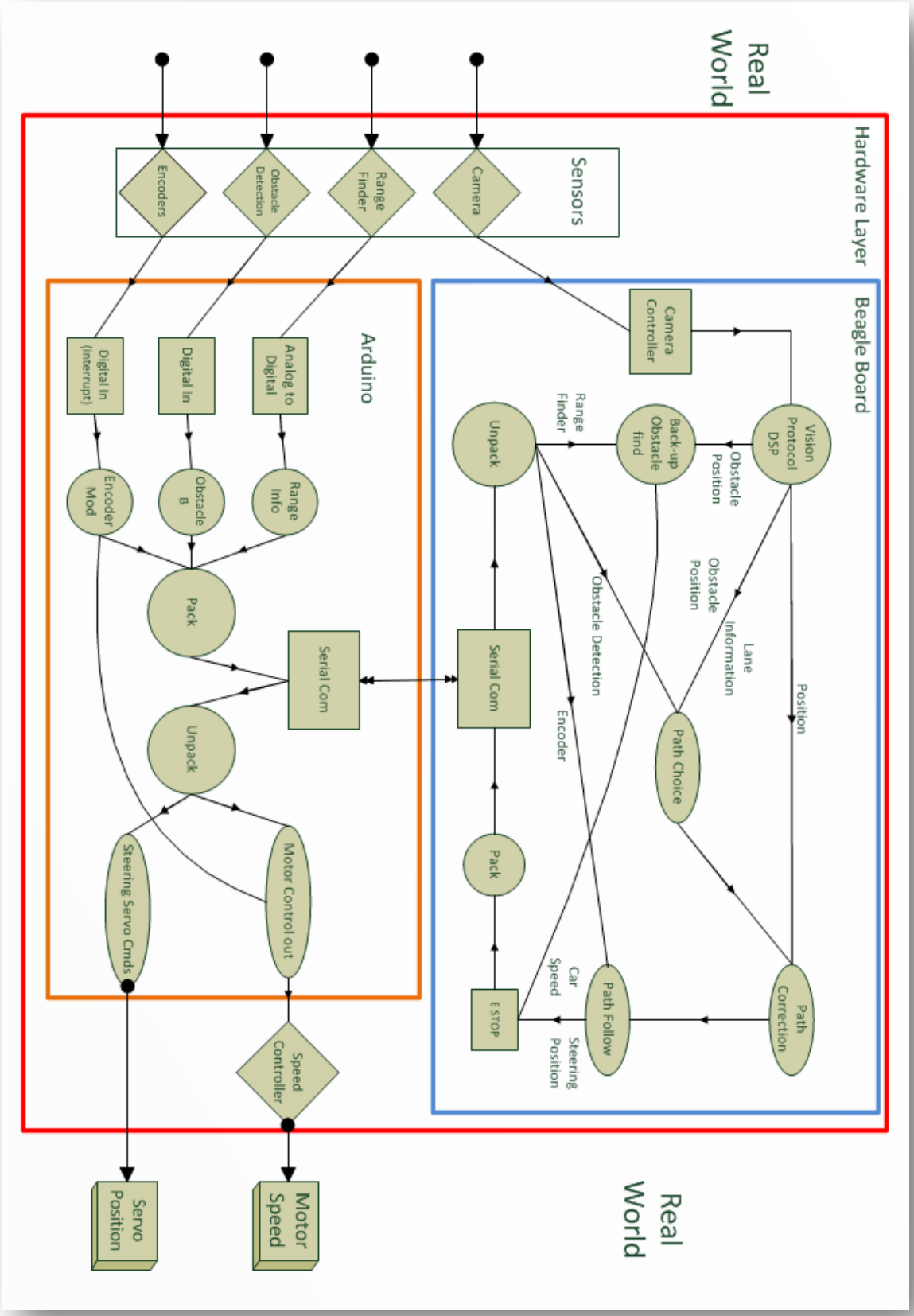


Figure 2: Overview of System

2.3.1 BeagleBoard-xM Component

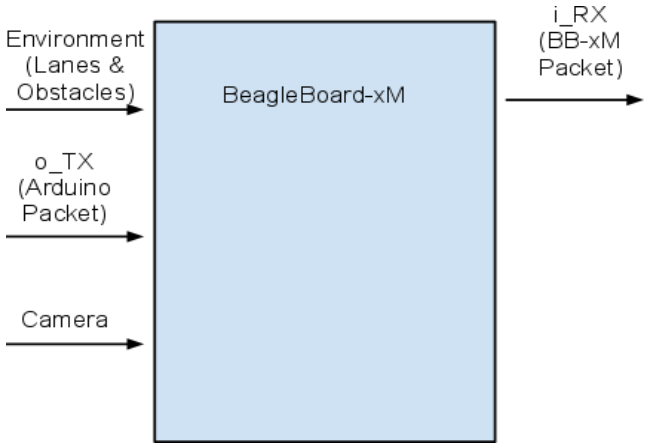


Figure 3: BeagleBoard-xM Context Diagram

2.3.2 Arduino Mega Component

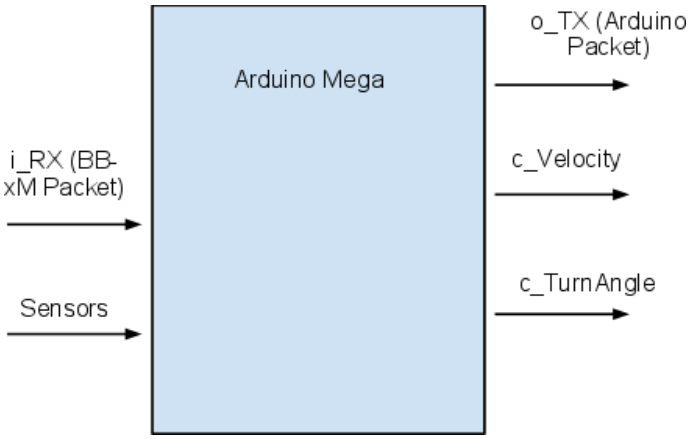


Figure 4: Arduino Context Diagram

2.4 Vehicle Chassis

The development team has decided to use the TT-01 Type-E chassis made by Tamiya as the base of the system.

The chassis is ideal since it, "features a longitudinally-mounted motor and reliable shaft-driven 4WD drivetrain. In addition to the durable bathtub frame, fiberglass-reinforced upper deck and gear covers increase rigidity. 4-wheel double wishbone suspension and 3-piece steering tie rod offer superb handling while front and rear differentials ensure power transfer efficiency..."[3]

Specifications: Length: 417mm, Width: 187mm, Height: 143mm[3]

With the above in mind the chassis meets the following requirements as described in the SRS:

1. The vehicle shall be able stay within one lane when traveling. (REF SRS 3.3.7)
    - The width of one lane is ~300mm so the vehicle having a width of 187mm fulfills this requirement and allows room for error.
2. The vehicle shall travel with smooth motion. (REF SRS 3.4.2)
    - The mechanical design of this chassis ensures complete and superb control over the vehicle at all times.
3. The vehicle shall use minimal power. (SRS REF 3.4.4)
    - The mechanical design of this chassis ensures power efficiency.
4. The vehicle shall appear aesthetically pleasing. (REF SRS 3.4.6)
    - Refer to Figure 5 below

Figure 5: TT-01 Type-E Chassis[4]

2.5 Sensor Component Placement

Below is a visual representation of the sensor placement on the vehicle.

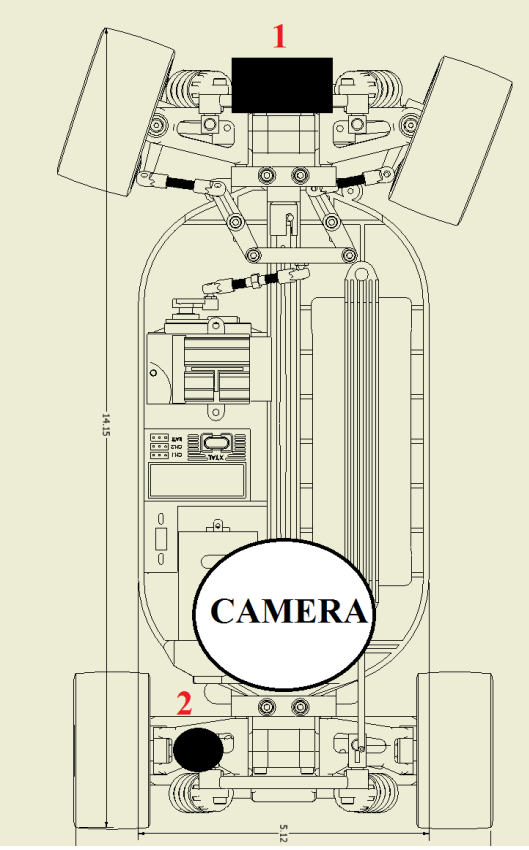


Figure 6: Sensor Component Placement

1. Front Range Finder Sensor (Parallax PING Ultrasonic range Sensor 4cm – 3m range) - Used as backup for camera obstacle detection.
2. Encoders (Hall Effect Sensor 55140-3H) - Used to detect speed of both rear wheels.
3. Camera (Playstation Eye 75 deg FOV) - Elevated to provide a greater field of view. Used primarily for lane following and obstacle avoidance.

In the initial stages of the design, the vehicle had ping sensors at the sides to detect passing obstacles. The team decided to remove these as it would add no benefit for the current scale of the vehicle and obstacles. The rear obstacle detectors have also been removed. The rationale is that since the client removed moving obstacles from his requirements, the time between detecting an obstacle, the current time, and speed will allow the vehicle to calculate if it is safely passed an obstacle or not. Likewise, this design decision also follows the system requirements of: K.I.S.S.

2.6 System State Diagram

The behaviour of the system can be found below as well as the states it can enter.

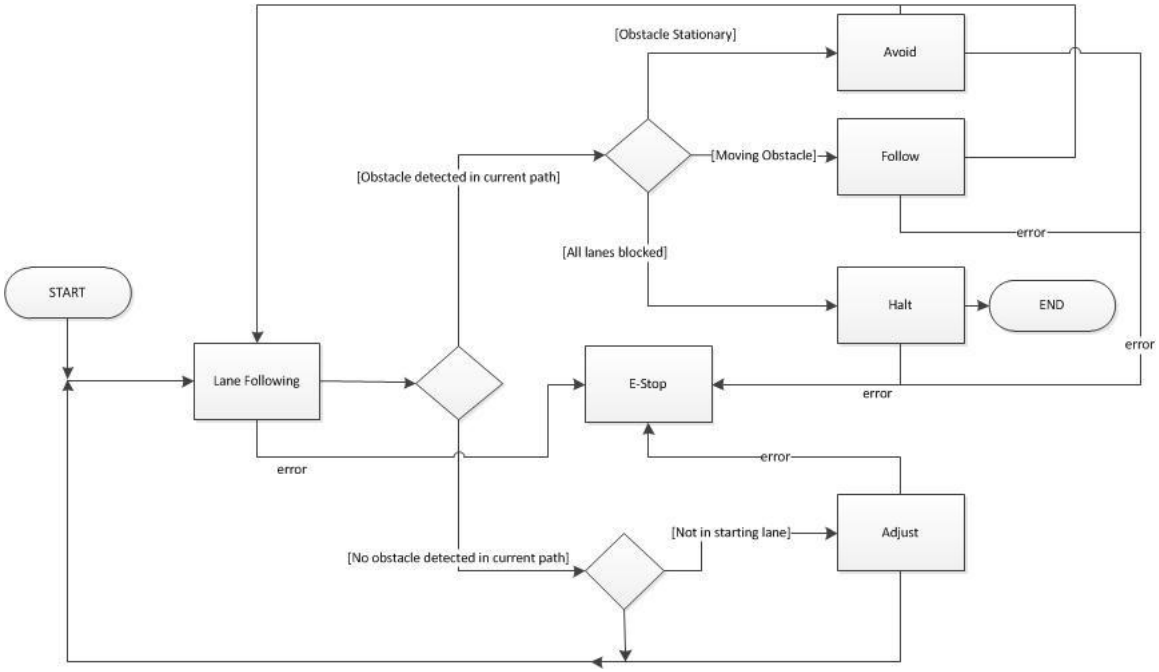


Figure 7: System State Diagram

State	Description
Lane Following	In this state the vehicle will progress (forward direction of vehicle's orientation) along the track while staying in between two lines (lane).
Avoid	In this state the vehicle will switch to the next available lane (ie. any unobstructed lane).
Halt	In this state the vehicle will come to a gradual stop.
Adjust	In this state the vehicle will move to a lane which is or closest to the initial lane. If it's not possible it will do nothing.
Follow	In this state the vehicle will follow a moving obstacle in front of it only if it's in the current lane. (Adaptive Cruise Control)

Table 2: State Descriptions

Please note in at all times the system is also in a sensing state (i.e. receiving information from sensors about environment).

3. SYSTEM COMPONENTS

The following are the system's components. Also included are the development team's considered hardware implementations for some of the components. As described in Section 2.3 there are two main components in this system. These are the Image Processing Unit (BeagleBoard-xM) and the Sensor/Control Processing Unit (Arduino Mega).

3.1 Image Processing Unit

The image processing unit is the heart of the system as it will be the primary monitoring device to feed the control system vital data regarding lane following and obstacle detection. Due to the heavy processing nature of image data it has been decided that the image processing shall take no longer than twice the processing time of the other units. This is may change after some initial testing.

3.1.1 Image Processing Unit Consideration

The development team has decided to use a BeagleBoard-xM because of its size, processing power and flexibility to program. The development team has decided whatever hardware we use it must fit well with the shape and form of the system and not look out of place. Due to its small size the BeagleBoard-xM will fit within the vehicle chassis with ease. Its square shape also helps matters making the internal architecture easy to deal with. Having the equivalent processing power as a low-end laptop or netbook gives the development team a lot of flexibility in terms of image processing and doing other tasks simultaneously while at the same time keeping the system weight to a minimum.

Specifications Summary [5]:

- Package on Package POP CPU/memory chip.
  - Processor TI DM3730 Processor - 1 GHz ARM Cortex-A8 core
  - 'HD capable' TMS320C64x+ core (800 MHz up to 720p @30 fps)
  - Imagination Technologies PowerVR SGX 2D/3D graphics processor supporting dual independent displays
  - 512 MB LPDDR RAM
  - 4 GB microSD card is supplied with the BeagleBoard -xM loaded with Angstrom.
- Peripheral connections
  - DVI-D (HDMI connector chosen for size - maximum resolution is 1280×1024)
  - S-Video
  - USB OTG (mini AB)

  - 4 USB ports
  - Ethernet port
  - MicroSD/MMC card slot
  - Stereo in and out jacks
  - RS-232 port
  - JTAG connector
  - Power socket (5 V barrel connector type)
  - Camera port
  - Expansion port
- Development
  - Boot code stored on the uSD card
  - Boot from uSD/MMC only
  - Alternative Boot source button.

3.2 Sensor and Control Processing Unit

The sensor and control processing unit is the primary interface between the sensors on the vehicle and control of the vehicle such as its speed and steering. It must be responsive and be able to meet very strict deadlines in order to meet all the requirements laid out for this system. The unit should also have a reasonable amount of I/O pins so that later in development if we feel we need to add a component such as a sensor it should not be much trouble.

3.2.1 Sensor and Control Processing Unit Consideration

The development team has decided to use an Arduino Mega 2560 because of its ease of use, size and amount of I/O pins. There are a lot of resources available on the internet regarding programming with Arduino boards so the learning curve is minimized for everyone involved. It is small and rectangular which helps reduce the complexity of the internal architecture of the system. Finally with 54 I/O pins we should have enough resources regardless of what reasonable system design of sensor configuration we use.

Specifications Summary [6]:

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

4. SYSTEM SUB COMPONENTS

4.1 Encoder

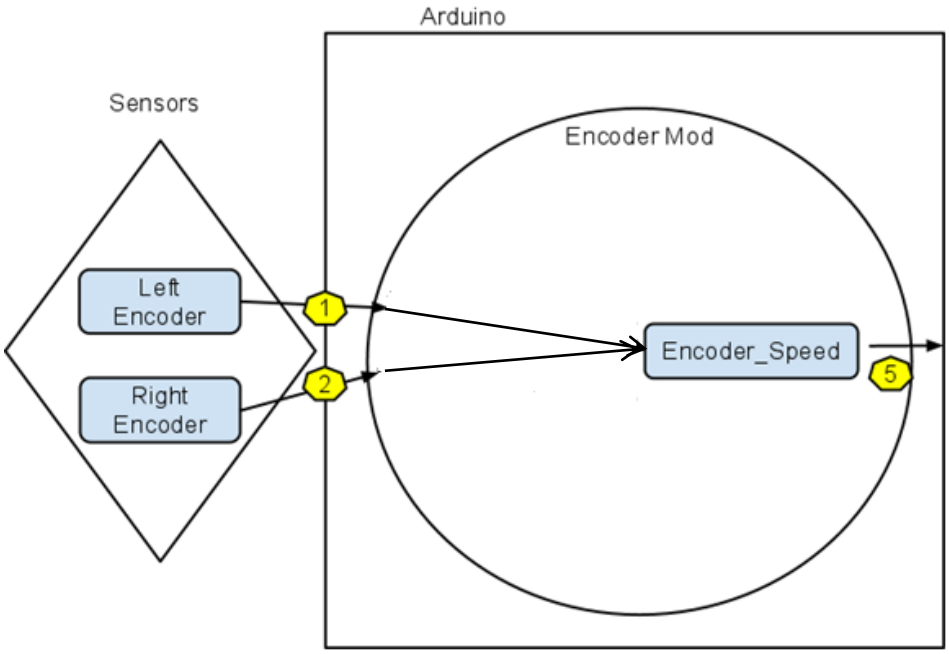


Figure 8: Encoder Component Overview

1. m_CurrentSpeedLeft
2. m_CurrentSpeedRight
5. o_CurrentSpeed

4.1.1 Encoder Left

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_CurrentSpeedLeft	Binary	[0,1]	N/A	

Outputs:

Output Name	Output Type	Range	Units	Comment(s)

i_CurrentLeftRPM	Int16	[0,500]	rot/s	The vehicle cannot physically go over 500 rot/s. Note rot = rotations.

Internal variables:

Internal Name	**Internal Type**	**Range**	**Units**	**Comment(s)**
Time	Int16	N/A	s	
$Time_{-1}$	Int16	N/A	s	Time at previous interrupt

Description:
This component will run based on an interrupt from Digital_In from m_CurrentSpeedLeft. The component will take the current time of the component and compare it to the last time it received a signal from the corresponding monitored variable and compute the number of rotations per second.

condition	i_CurrentLeftRPM
m_CurrentSpeedLeft	60/(Time- Time-1)
~m_CurrentSpeedLeft	No Change

Initialization:
i_CurrentLeftRPM = 0 rot/s
Time-1 = 0 s

Rationale:
It shall be assumed that the vehicle is not travelling when first placed within the environment; this is a reasonable assumption as specified by the client.

Timing Requirements:
1. Operate at a speed such that no interrupts on either the m_CurrentSpeedLeft are missed.
2. Operate at a speed to meet the deadline set by the Pack module of the Arduino system.

Requirements fulfilled:

1. The vehicle shall travel as fast as possible around the track. (REF SRS 3.4.1)
2. The vehicle shall only be allowed to travel in the forward direction. (REF SRS 3.3.1)

Rationale:

This component will be a crucial part in determine the speed of the vehicle. The speed of the vehicle is crucial in order to be able to travel as fast as possible (REF SRS 3.4.1) and to determine if we are only traveling in the forward direction (REF SRS 3.3.1). This module is crucial due to the differential drive of the system, both left and right tire speed will be necessary to determine the speed of the vehicle.

Build Issues:
- Missed interrupt
- Failure of hardware components

4.1.2 Encoder Right

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_CurrentSpeedRight	Binary	[0,1]	N/A	

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
i_CurrentRightRPM	Int16	[0,500]	rot/s	The vehicle cannot physically go over 500 rot/s. Note rot = rotations.

Internal variables:

Internal Name	Internal Type	Range	Units	Comment(s)
Time	Int16	N/A	s	
Time-1	Int16	N/A	s	Time at previous interrupt

Description:
This component will run based on an interrupt from Digital_in from m_CurrentSpeedRight. The component will take the current time of the component and compare it to the last time it received a signal from the corresponding monitored variable and compute the number of rotations per second.

condition	i_CurrentRightRPM
m_CurrentSpeedRight	60/(Time- Time-1)
~m_CurrentSpeedRight	No Change

Initialization:
i_CurrentRightRPM = 0 rot/s
Time-1 = 0 s

Rationale:
It shall be assumed that the vehicle is not travelling when first placed within the environment; this is a reasonable assumption as specified by the client.

Timing Requirements:
1. Operate at a speed such that no interrupts on either the m_CurrentSpeedLeft are missed.
2. Operate at a speed to meet the deadline set by the pack module of the Arduino system.

Requirements fulfilled:

1. The vehicle shall travel as fast as possible around the track. (REF SRS 3.4.1)
2. The vehicle shall only be allowed to travel in the forward direction. (REF SRS 3.3.1)

Rationale:
This component will be a crucial part in determine the speed of the vehicle. The speed of the vehicle is crucial in order to be able to travel as fast as possible (REF SRS 3.4.1) and to determine if we are only traveling in the forward direction (REF SRS 3.3.1). This module is crucial due to the differential drive of the system, both left and right tire speed will be necessary to determine the speed of the vehicle.

Build Issues:

- Missed interrupt
- Failure of hardware components

4.1.3 Encoder Speed

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
i_CurrentLeftRPM	Int16	[0,500]	rot/s	
i_CurrentRightRPM	Int16	[0,500]	rot/s	

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
o_CurrentSpeed	Int16	[0,500]	cm/s	

Internal variables:

Internal Name	Internal Type	Range	Units	Comment(s)
CurrentLeftRPM	Int16	[0,500]	rot/s	
CurrentRightRPM	Int16	[0,500]	rot/s	

Description:
This component will run based on an interrupt from Encoder_Left or Encoder_Right. It will update the corresponding internal variables and calculate the average speed.

		o_CurrentSpeed
i_CurrentLeftRPM interrupt	i_CurrentLeftRPM == CurrentLeftRPM	[(CurrentLeftRPM +CurrentRightRPM)/2]*k_Circumference
i_CurrentRightRPM interrupt	i_CurrentRightRPM == CurrentRightRPM	[(CurrentLeftRPM +CurrentRightRPM)/2]*k_Circumference
No interrupt		No Change

Initialization:
o_CurrentSpeed = 0 cm/s
CurrentRightRPM = 0 rot/s
CurrentLeftRPM = 0 rot/s

Rationale:
It shall be assumed that the vehicle starts in a state of 0 cm/s until otherwise determined.

Timing Requirements:
1. Operate at a speed such that no interrupts from Encoder_Left or Encoder_Right are missed.

2. Operate at a speed to meet the deadline set by the Pack module of the Arduino system.

Requirements full filled:

1. The vehicle shall travel as fast as possible around the track. (REF SRS 3.4.1)
2. The vehicle shall only be allowed to travel in the forward direction. (REF SRS 3.3.1)

Rationale:
This component will be a crucial part in determining the speed of the vehicle. The speed of the vehicle is crucial in order to be able to travel as fast as possible (REF SRS 3.4.1) and to determine if we are only traveling in the forward direction (REF SRS 3.3.1). This component is crucial as it takes in the RPM of both left and right tires and computers the current speed in cm/s.

Build Issues:

- Missed interrupt
- Failure of hardware components

4.1.4 Encoder Sensor Considerations

The project team has identified several possible hardware options for the encoder that will meet requirement and design specifications. The first two options are forms of rotary encoders, the first being a Hall Effect sensor mounted by each rear wheel with a small magnet attached to the wheel to record when a full revolution of the wheel has been completed. The second is a system on a chip rotary encoder which will provide higher accuracy and all the necessary smoothing and A/D conversions. Below is a diagram depicting the system on the chip:

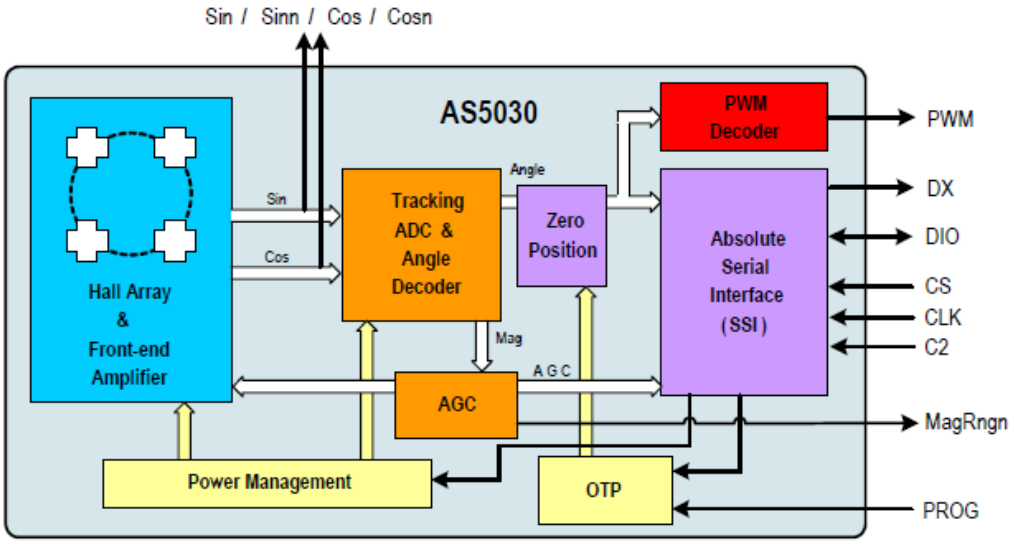


Figure 9: Rotary Encoder Example [7]

The last possible hardware choice the team is reviewing is a quadrature encoder attached to the drive shaft.

The project team has decided upon using a Hall Effect sensor (55140-3H Flange Mount Hall Effect Sensor). The other options were not suitable for our problem since a very accurate reading was not required, they were too expensive and mounting them onto the chassis would be very difficult. With the Hall Effect sensor we are able to easily mount it to the chassis and produce a reliable RPM reading and at the same time keep costs to an absolute minimum compared to the other solutions.

The sensor's output voltage depends on the presence of a magnetic field. Currently the configuration consists of a single D7.5x2.5mm neodymium disc shaped rare earth magnet, with a field strength of 142 Guass at 10mm from sensor[11], fixed onto the outer edge of a back tire. The sensor is placed on the wheel hub assembly. As the magnet

rotates with the tire it passes the sensor with a distance of approximately 10mm. When the sensor passed directly in front of the sensor the sensor outputs a constant 0V.

4.2 Motor Control Out

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
d_Velocity	Int16	0 cm/s to TBD	cm/s	Requested speed coming from the serial com.
o_CurrentSpeed	Int16	[0,500]	cm/s	Speed information from encoders module

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
x_Velocity	PWM signal	0% to TBD	Duty Cycle	PWM signal going to the speed controller

Internal Constants:

Internal Name	Internal Type	Range	Units	Comment(s)
Vstep_Up	Int16	5 cm/s	cm/s	This constant will be used to increase the speed in discrete steps.
Vstep_Down	Int16	-5 cm/s	cm/s	This constant will be used to decrease the speed in discrete steps.

Description:
This module will map the desired value of speed provided through the serial com, to an actual speed which will be monitored through the encoder. It will increase in the PWM signal in discrete steps until the speed in reached.

Initialization:
This module will initialize at 0% duty cycle until it is assigned a speed from the serial communication.

Timing Requirements:
This module will perform according to system specification in terms of refresh rates. However, it may go faster, depending of the limitation imposed by the feedback control.

Requirements Fulfilled:

- The vehicle shall only be allowed to travel in the forward direction

Build Issues:
A possible issue with this module is the response time to achieve the desired speed.

4.3 Steering Servo Command

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
d_TurnAngle	Int16	[-60,60]	degrees	The change in wheel position requested by the serial com.

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
c_TurnAngle	PWM signal	0% to TBD	duty cycle	This signal will feed directly into the servo.

Internal Constants:

Internal Name	Internal Type	Range	Units	Comment(s)
V_Step	Int16	TBD	degrees	Discrete change in servo position to meet desired change.

Description:
This module will receive a desired change in the wheels position and map it to a PWM signal that will tell the servo to move a certain amount.

Initialization:
This module should initialize sending a zeroing signal to the servo, making sure the wheels are completely straight.

Timing Requirements:
This module should be subjected to the same timing requirements as the full system in terms of refresh rates.

Requirements Fulfilled:

- The vehicle shall have the ability to change lanes
- The vehicle shall maintain control at all times when traveling.
- The vehicle shall avoid obstacles.

Build Issues:
The possibility of losing track of the servo's position because there is no real feedback.

4.4 Camera

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_CameraImage	Normalized decimal (3D Array [] [] [])	Size - mxn, depending on camera. xo number of colour channels per pixel. Range - 0-1	Colour %	Double array of percent for each pixel.

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
m_ObsCheckX X = [0, 3]	Binary	[0,1]	N/A	Determine whether lane X has an obstacle
m_ObsDistX[n] X = [1, 3] n = 0, 1, 2, 3,...	Int16 Array	[0,100]	cm	Distance between system and n obstacle(s) in lane X
m_LanePointsX[n] X = [1,3] n = [0,100]	Position	[0,40]	cm	Position of n mid points detected for lane X. n varies.
m_CurrentLane	Integer	[0,3]	N/A	Which lane we are in
m_DistanceToCenter	Vector	[-5,5]	cm	Vector direction away from center for current lane.
e_DefaultLane	Int16	[0,3]	cm	Detect the lane the system starts in

Internal variables:

Internal Name	Internal Type	Range	Units	Comment(s)
m_FixedImage	Binary (2D Array [] [])	[0,1]	N/A	Camera image post-processing into b/w image.

Description:
The camera module is responsible for detecting lanes and obstacles in front of the car in enough time to avoid collisions. The camera module does three steps of processing. First the module sharpens contrast by going over each pixel and preforming the following comparison. This will produce a single black and white output image. The calculation computes the Y factor from the YUV colour space conversion from RGB.

0.299*Pr + 0.114*Pb + 0.587 * Pg	**Out Pixel**
>=0.5	1
<0.5	0

Next lanes are found by searching across the image for white lines. Blobs are ignored as those are likely obstacles. This information will then be transformed into a line for each found lane with an approximation of its route. This line is broken down into points k_PointsDistance(1cm) apart for processing down the path.

Finally a search is performed for the blobs, and their location along each line is found and reported. Its location on the map is referenced against m_LanePoints to determine its lane, and then based on its index into it will determine distance. When an obstacle is detected in a lane, m_ObsCheckX will marked appropriately.

Initialization:
All outputs output 0 during initialization until data appears.

Timing Requirements:
New data must appear every 29.3ms. Please keep in mind this value is very likely to change after some hardware testing.

Build Issues:
This algorithm will be expensive in computation power, and will difficult to implement.

4.4.1 Camera Considerations

The final design consists of a Playstation Eye.

Specifications:

Manufacturer: SCEI
Connectivity: USB 2.0 (type-A)
Resolution: 640x480 pixels @ 60Hz
320x240 pixels @ 120Hz
Field of View (FOV): 56^{o} or 75^{o}
Dimensions: 80mm x 55mm x 65mm

The decision to use this camera is as follows. First, the 75^{o} FOV setting allows the system to track all lanes at all times even on the turns of the track and secondly it's relatively low cost ($20.00). It also allows us to poll the camera fast enough.

4.5 Camera DeBounce

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_LanePointsX[n] X = [1,3] n = [0,100]	Position	[0,40]	cm	Position of n mid points detected for lane X. n varies.

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
m_LanePointsX[n] X = [1,3] n = [0,100]	Position	[0,40]	cm	Position of n mid points detected for lane X. n varies.

Internal variables:

Internal Name	Internal Type	Range	Units	Comment(s)
m_PreviousLanePointsX[n] X = [1,3] n = [0,100]	Position	[0,40]	cm	Position of n mid points detected for lane X. n varies. This is what was previously sent

Description:
This module uses the previous lane points to interpolate where the new points should be. Any points that are missing from the camera will be re-added here to attempt to keep information relevant.

Initialization:
m_PreviousLane will be empty, and the algorithm will output this empty value until data appears on the input.

Timing Requirements:
New data must appear 3ms after receiving the new points. Please keep in mind this is very likely to change after hardware testing.

Build Issues:
Data storage for these points is important.

4.6 Front Range Finder

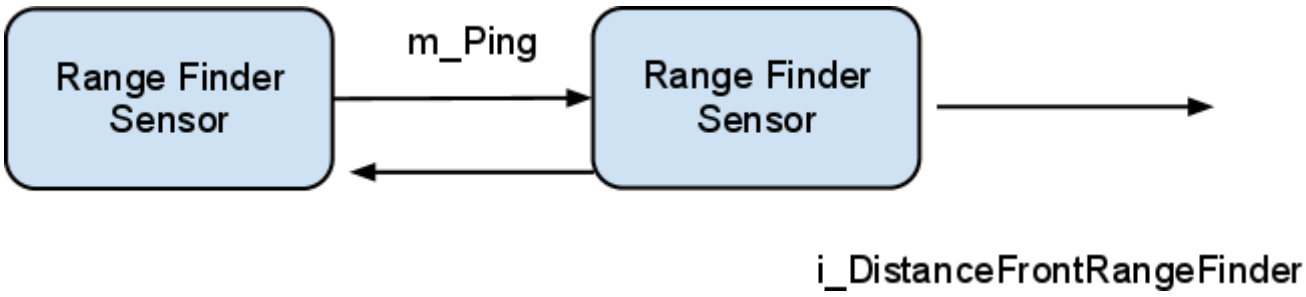


Figure 14: Front Range Finder Component Overview

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_Ping	Binary	[0,1]	N/A	

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
i_DistanceFrontRangeFinder	Int16	[2,300]	cm	

Description:
Commercial range finder sensors work by sending a high pulse to the sensor for a period of time in order to trigger the sensor to send out a signal and return the result.

Initialization:
i_DistanceFrontRangeFinder = 2 cm

Rationale:
We shall assume that on start-up that an obstacle is right in front of the vehicle until it's determined otherwise.

Timing Requirements:
Commercial Range Finder sensors have a timing response of approximately 18.5ms. Likewise, the time it requires to send a high signal to the range finder sensor also has to be considered in the timing response.

Requirements fulfilled:

1. The vehicle shall be able to decide to stop when no further progress is possible. (SRS REF 3.3.16)
2. The vehicle shall not make physical contact with another object on the track. (SRS REF 3.3.11)
3. The vehicle shall have the ability to detect obstacles in each lane in the direction it is traveling. (SRS REF 3.3.13)

Rationale:
The main hazard in the environment for the vehicle is obstacles in front of the vehicle in its current lane. This module will allow for the vehicle to detect obstacles in its current lane (SRS REF 3.3.13). Likewise, the output of this module will be a crucial component in being able to determine when no further progress is possible (SRS REF 3.3.16) as well as preventing physical contact with obstacles (SRS REF 3.3.11).

Build Issues:
This module is technically possible.

4.6.1 Front Range Finder Sensor Considerations

The design team has decided in using an ultrasonic sensor for the front range finder sensor. The rationale behind this decision is that an ultrasonic sensor provides the timing requirements and the range needed to meet the requirements of this project. In comparison to other range finder sensors such as IR, an ultrasonic sensor provides superior protection against noise and lighting changes in the environment. Below is a schematic showing a commercially available ultra-sonic sensor connected to an Arduino. This demonstrated how the hardware setup is technically feasible. Note, the below schematic is not a finalized design for the Ultra Sonic Sensor.

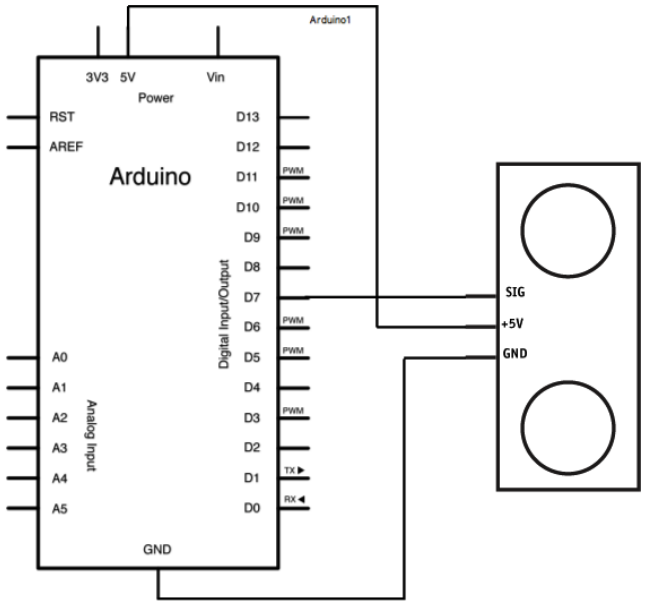


Figure 15: Front Range Finder Sensor Example

The final design consists of a Parallax PING Ultrasonic Range sensor (2cm-3m range). This ultrasonic range sensor works very well with the Arduino MEGA and only requires 3 pins since it has a single shared I/O pin for transmitting and receiving a ping.

4.7 Path Choice

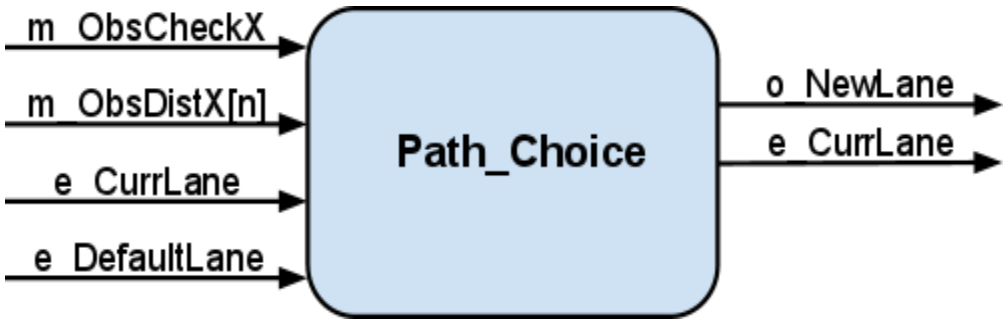


Figure 16: Path Choice Overview

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_ObsCheckX X = [0, 3]	Binary	[0,1]	N/A	Determine whether lane X has an obstacle
m_ObsDistX[n] X = [1, 3] n = 1, 2, 3,...	Int16 Array	[0, 100]	cm	Distance between system and obstacle n in lane X
e_DefaultLane	Int16	[0,3]	N/A	This is the very first lane the system initializes its operation.

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
o_NewLane	Int16	[0,3]	N/A	The new lane the system must change to. See e_CurrLane for additional details.

Internal Variables:

Internal Name	Internal Type	Range	Units	Comment(s)
e_CurrLane	Int16	[0,3]	N/A	Current lane vehicle is situated in. Lane 0 does not exist but is an initialization stage where after

				the first execution frame the variable shall be a value between [1,3]

Description:
The PathChoice module is responsible for determining whether the system must deviate from its current path depending on the oncoming conditions of the track. The module uses lane information, obstacle position, and obstacle detection data to create an output stating the new lane the vehicle should move to. If the value of the new lane is equal to the value of the current lane then no lane change shall occur.

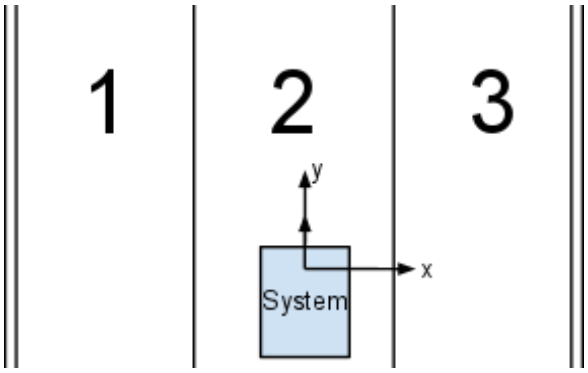


Figure 17: Path Choice

Below is a sample of data referring to when the default lane and current lanes are both Lane 1. Please refer to Appendix A 4.1 Path Choice Table section at the end of this document for a complete set of scenarios regarding the lane changing algorithm.

Note: Distances may be tentative depending on further testing of hardware.

e_DefaultLane	**e_CurrentLane**	**m_ObsCheck[X]**	**m_ObsDis1[1]**	**m_ObsDist2[1]**	**m_ObsDist3[1]**	**o_NewLane**
1	1	[0,0,0]	>50	>50	>50	1
1	1	[1,0,0]	[20,50]	>50	>50	2
1	1	[1,0,0]	<20	>50	>50	2
1	1	[1,1,0]	[20,50]	[20,50]	>50	2
1	1	[1,1,0]	[20,50]	<20	>50	1
1	1	[1,1,0]	<20	[20,50]	>50	2
1	1	[1,1,0]	<20	<20	>50	1
1	1	[1,1,1]	[20,50]	[20,50]	[20,50]	1
1	1	[1,1,1]	<20	[20,50]	[20,50]	2
1	1	[1,1,1]	<20	<20	[20,50]	1

1	1	[1,1,1]	<20	<20	<20	1
1	1	[1,1,1]	[20,50]	[20,50]	<20	1
1	1	[1,1,1]	[20,50]	<20	<20	1
1	1	[1,1,1]	[20,50]	<20	[20,50]	1
1	1	[1,1,1]	<20	[20,50]	<20	2
1	1	[0,1,1]	>50	[20,50]	[20,50]	1
1	1	[0,1,1]	>50	<20	[20,50]	1
1	1	[0,1,1]	>50	<20	<20	1
1	1	[0,1,1]	>50	[20;50]	<20	1
1	1	[0,0,1]	>50	>50	[20,50]	1
1	1	[0,0,1]	>50	>50	<20	1
1	1	[1,0,1]	[20,50]	>50	[20,50]	2
1	1	[1,0,1]	<20	>50	[20,50]	2
1	1	[1,0,1]	<20	>50	<20	2
1	1	[1,0,1]	[20,50]	>50	<20	2
1	1	[0,1,0]	>50	[20,50]	>50	1
1	1	[0,1,0]	>50	<20	>50	1

The current distances were chosen to take into account that it is theoretically possible to change lanes between two close obstacles as long as there is approximately 30 cm of clearance between them, as illustrated below.

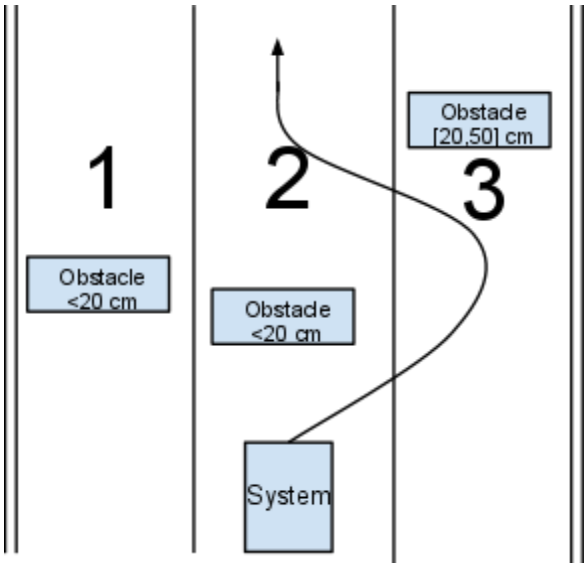


Figure 18: Path Choice Behaviour

If only the 50 cm barrier had been used, all three obstacles would fall under the <50 cm range and the system will never detect the opportunity to go in between the lanes. This situation is unlikely as our system will most likely start in a safer state where the obstacles are farther away but it is still important to take into account if this situation occurs right after a lane change. In most cases this situation would be detected much earlier.

Initialization:
m_ObsCheckX = 0
m_ObsDistX[n] = NULL
e_DefaultLane = 0
e_CurrLane = e_DefaultLane
o_NewLane = 0

Timing Requirements:
The module shall complete the transfer from inputs to outputs and make a decision on changing lanes within 32ms, or 2 frames of execution as the image processing of the camera device is assumed to be running at 30 frames per second (fps), half the speed of the rest of the system of 60 fps.

Requirements fulfilled:

- The vehicle shall have the ability to detect obstacles in each lane in the direction it is traveling.
- The vehicle shall travel in its starting lane whenever possible.

Rationale: As stated by the client it is preferable that the system stay in its starting lane as much as possible. The logic behind the lane changes does take this constraint into

account. The logic behind this component takes into account all situations of obstacle layout and makes decisions based on the aforementioned requirements.

Build Issues:

- If any one of the input values is not refreshed to its current state the output may be corrupted or out-dated.
- The ranges will have to be modified to reflect the capability of the hardware and any interference or discrepancies between the imaging and sensing protocols.
- Target refresh rate of 32ms may not be feasible with image processing. Trade-off between processing time and accuracy.
- Lane decision not made on time may cause system to collide with an obstacle. Strict deadlines must be met for this component to function properly.

4.8 Path Correction

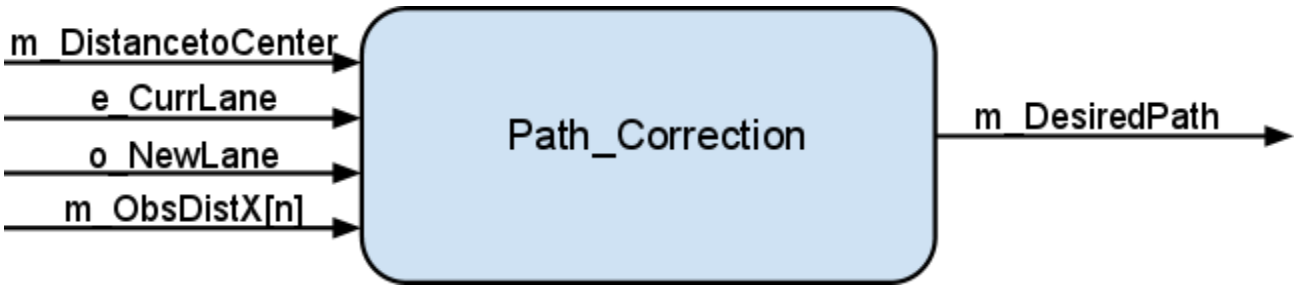


Figure 19: Path Correction Overview

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_DistancetoCenter	Int16	[-5,5]	cm	Distance of center of system to the middle of the lane
m_ObsDistX[n] X = [1,3] n = 1, 2, 3	Int16 Array	[0,100]	cm	Distance between the system and oncoming n obstacles in lane X
e_CurrLane	Int16	[0,3]	N/A	The lane the system is currently lane.
o_NewLane	Int16	[0,3]	N/A	The lane the system has decided to be after obstacle detection

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
o_DesiredPath	e_Path	[FORWARD, LEFT[1,10], RIGHT[1,10], STOP]	N/A	The next state of motion the system shall take

Description:
If the system decides to stay within the same lane the purpose of this component is to correct the path of the system within a lane so that the system is travelling in the middle of the lane for the majority of the time. However if it is decided the system must change lanes from Path_Choice then the component determines what direction and angle the system will move towards. As long as the midpoints of the lane are given they will follow the path of the lanes and allow the vehicle to travel along the trajectory of curves as well.

Condition	m_ObsDistX[n]	Distance to lane center	m_DesiredPath
o_NewLane - e_CurrLane = 0	>20	m_DistancetoCenter>0	RIGHT[m_DistancetoCenter+1]
	<20	DON'T CARE	STOP
	>20	m_DistancetoCenter<0	LEFT[abs(m_DistancetoCenter) +1]
o_NewLane - e_CurrLane = 1	DON'T CARE	DON'T CARE	RIGHT[8]
o_NewLane - e_CurrLane = -1	DON'T CARE	DON'T CARE	LEFT[8]

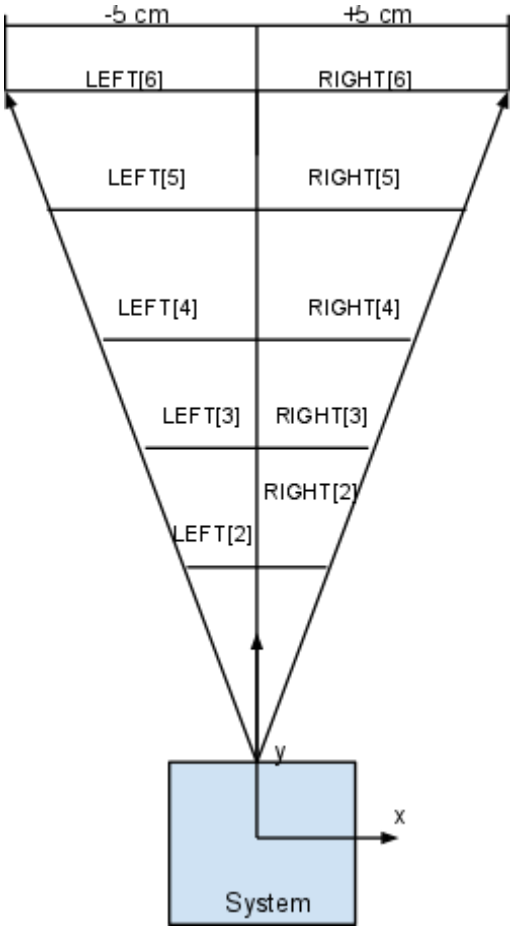


Figure 20: Motion of system within a single lane

Initialization:
m_ObsDistX[n] = NULL
e_CurrLane = o_NewLane = 0
m_DesiredPath = NULL
m_DistancetoCenter = NULL

Timing Requirements:
All output shall be processed at a periodic rate of approximately 32ms running at 30 frames per second.

Requirements Fulfilled (Refer to SRS for full description):

1. The vehicle shall have the ability to change lanes.
2. The vehicle shall stay within the boundaries of the track when traveling.
3. The vehicle shall have a lane following ability
4. The vehicle shall be able to avoid obstacles.
5. The vehicle shall only cross one lane at a time
6. The vehicle shall be able to decide to stop when no further progress is possible.
7. The vehicle shall only be allowed to travel in the forward direction.

Build Issues:

- Anomalies in lighting conditions may cause Camera component to send corrupted data related to the midpoint of a lane and obstacle distances.
- Current state of algorithm may or may not be sufficient for the situation of when there is an obstacle in a curve.
- Input data is not synchronized to enter the module at the same time.
- Timing conditions not being met to finalize the turn decision.
- Vehicle constantly oscillating within lane if component is implemented incorrectly.

4.9 Path Follow

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
e_DesiredPath	{e_Path, Int16}	[FORWARD, LEFT[1,10] , RIGHT[1,10] , STOP]	none	The desired path to be followed. This is a high level abstraction for the chosen path. It contains the direction and an integer value relating to the magnitude of the direction. Forward and Stop are Null magnitude.
m_Velocity	Int16	[0.00, 3.00]	m/s	A signal monitoring the speed (m/s) of the system in relation to the environment. The max speed is to be determined

Outputs:

Output Name	**Output Type**	**Range**	**Units**	Comment(s)
c_Velocity	Int16	[0.00, 300.00]	cm/s	The controlled variable, the system speed
c_TurnAngle	Int16	[-60, 60]	Degrees	The turning angle of the wheels of the system. The vehicle hardware limits the angle.

Description:
The Path Follow module is responsible for directing the vehicle to follow the specified path provided by the path Correction Module. This module monitors and controls the speed of the vehicle and uses information provided by the Path Correction. This information is used to provide the high level control of the vehicle which includes changing the vehicles direction and scaling the speed according the turning angle. The output of this module is then packed and sent over the serial communication.

Condition	c_Velocity	c_TurnAngle
e_DesiredPath == FORWARD	k_VelocityMax	0
e_DesiredPath == LEFT[x]	k_VelocityMax - 20*x	-60 * x/10
e_DesiredPath == RIGHT[x]	k_VelocityMax - 20*x	60 * x/10
STOP	0	0

Initialization:
m_DesiredPath = Null
m_Velocity = 0
c_Velocity = 0
c_TurnAngle = 0

Timing Requirements:
The module shall complete the transfer from inputs to outputs and provide high level control of the vehicles movement with a delay no greater than 16 ms, or a frame of execution.

Requirements Fulfilled (Refer to SRS for full descriptions):
- The vehicle shall maintain control at all times when traveling.
- The vehicle shall have the ability to stop.
- The vehicle shall travel as fast as possible around the track.
- The vehicle shall travel with smooth motion.

Build Issues:
Determining actual turning angles that are appropriate for specific speeds.

4.10 Backup Obstacle Detection

Inputs:

Input Name	Input Type	Range	Units	Comment(s)
m_ObsDist	Int16	[0, 100]	cm	The distance between the vehicle and an approaching obstacle in the current lane.
d_DistanceUltraSonic	2 bytes	[0, 65535]	none	The raw distance measurement produced by the ultra sonic sensor. Is used as a backup.
m_Impact	Bool	[True, False]	none	Determine if system has collided with another object in the environment

Outputs:

Output Name	Output Type	Range	Units	Comment(s)
e_DesiredPath	e_Path	STOP	none	This is used to send a STOP signal to the Path Follow module. The range is only STOP because we are not trying to avoid the obstacle; we want to stop before a collision.

Internal variables:

Internal Name	Internal Type	Range	Units	Comment(s)
distRangeFinder	Int16	[0, 100]	cm	Distance from vehicle to obstacle, provided by the range finder and converted to cm

Description:
The Backup obstacle detection module is responsible for detecting any undetected obstacles. This module monitors the obstacle distance information from the camera and the range finder. Information from the range finder is directly sent to this module it is received as a raw distance value in the form of bits rather than a boolean value.

condition	e_DesiredPath
m_Impact == True	STOP
m_ObsDist < 5 cm	STOP
distRangeFinder < 5 cm	STOP

Initialization:
m_ObsDist = 0
d_DistanceUltraSonic = 0
m_Impact = False
c_Velocity = 0

Timing Requirements:
The module shall complete the transfer from inputs to outputs and control the vehicles movement with a delay no greater than 16ms, or a frame of execution.

Requirements Fulfilled (Refer to SRS for full descriptions):

- The vehicle shall be able to avoid obstacles.
- The vehicle shall have the ability to stop.
- The vehicle shall be able to decide to stop when no further progress is possible.

Build Issues:

- Making sure there are no false detections of obstacles.
- Detecting obstacles if and when the range finder experiences interference.

5. ARDUINO MEGA BEAGLEBOARD-xM INTERFACE

To communicate between the Arduino and the BeagleBoard-xM, serial communication has been considered. The following describes a basic overview of functionality.

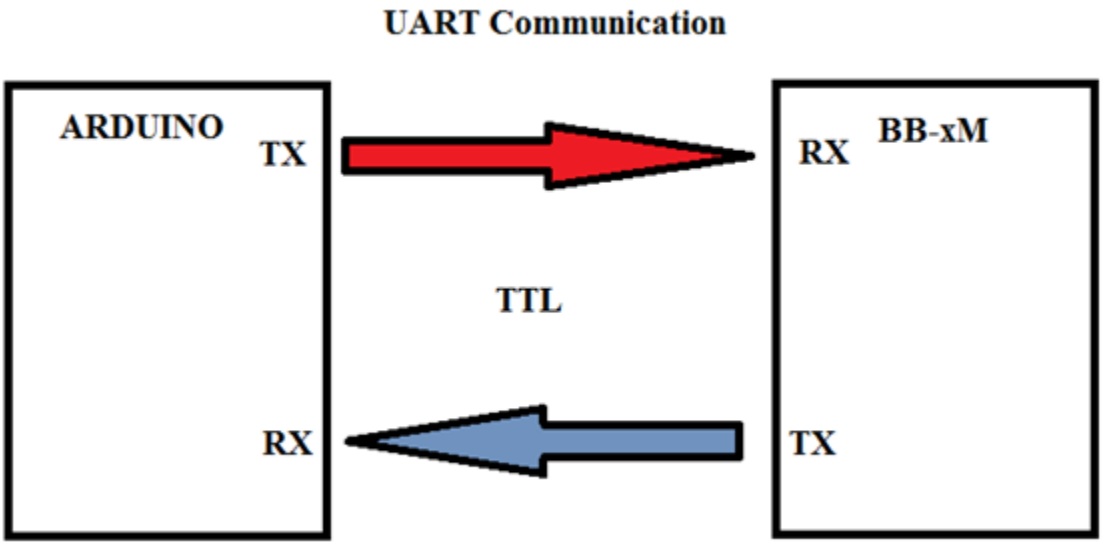


Figure 21: Interface Overview

5.1 Communications from BeagleBoard-xM to Arduino Mega

Description:
This module uses the outputs from module 4.10 Path Follow. These outputs are 'packed' into an oTX packet and sent to the Arduino over serial. It will then be 'unpacked' in a suitable manner to identify parts of the packet.

5.1.1 Structure of Serial Packet (oTX/oRX)

Name	Size (bytes)	Start Byte Number	Description
HEADER	3	0	Header (3 bytes of 0xFF)
DATA	4	3	Raw data to Arduino
PARITY	1	7	Error detection byte
STOP	2	8	Stop (2 bytes of 0xFF)

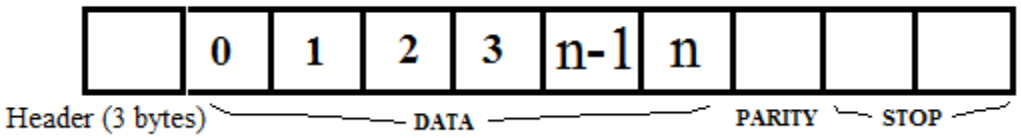


Figure 22: Packet Configuration

5.1.2 Data Contents of oTX/oRX

Name	Size (bytes)	Start Byte Number	Description
d_Velocity	2	0	Set velocity
d_TurnAngle	2	2	Set turn angle

Initialization of Data:

Name	Size (bytes)	Initial Value
d_Velocity	2	0
d_TurnAngle	2	90

Timing Requirements:
Data must be sent and received in a timely manner that meets timing requirements in SRS.

5.2 Communications from Arduino Mega to BeagleBoard-xM

Description:
This module uses the outputs from modules 4.1.3 Encoder Speed, 4.6 Rear Range Finders and 4.7 Front Range Finder. These outputs are ‘packed’ into an iTX packet and sent to the BeagleBoard-xM over serial. It will then be ‘unpacked’ in a suitable manner to identify parts of the packet.

5.2.1 Structure of Serial Packet (iTX/iRX)

Name	Size (bytes)	Start Byte Number	Description
HEADER	3	0	Header (3 bytes of 0xFF)
DATA	1	3	Raw data to BeagleBoard-xM
PARITY	1	4	Error detection byte
STOP	2	5	Stop (2 bytes 0xFF)

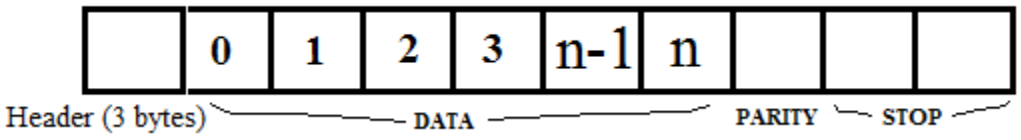


Figure 23: Packet Configuration

5.2.2 Data Contents of iTX/iRX

Name	Size (bytes)	Start Byte Number	Description
d_DistanceUltraSonic	2	0	Distance of obstacle in front of vehicle

Initialization of Data:

Name	Size (bytes)	Initial Value
d_DistanceUltraSonic	2	2

Timing Requirements:

Data must be sent and received in a timely manner that meets timing requirements in SRS.

6. SYSTEM BEHAVIOUR

6.1 Normal Operation

Under normal operating conditions the vehicle shall be able to meet the requirements of being able to drive within the confines of a lane and if need be, to change lanes when an obstacle is detected in the current lane. Within the design itself, all timing constraints outlined in the SRS will be met by their deadlines.

6.2 Undesired Event Handling

In the case that an error occurs in the operation of the system an “Emergency Stop” state has been added to make sure the vehicle does not proceed any further to prevent any damage to the environment or the system itself. The “Emergency Stop” state will prevent the following undesired events:
1. A foreign object is detected by the sensors (such as a person entering the environment and walking in front of the vehicle), the Emergency Stop will prevent the undesired event of crashing into the foreign object.
2. A sensor stops working as desired
3. The operating system crashes or has a bug

In case of small inclines on the track a feedback system has been implemented on the wheels to ensure the vehicle is always travelling at the desired speed and turn angle.

7. MIS & MID

7.1 Module: Encoder

7.1.1 MIS

Determines the current RPM of the specified wheel of the vehicle based on the pin number the encoder is attached too and adjusts the motor RPM to adjust from any deviations from the desired RPM.

Interface

Uses

None

Type

O_CurrentRPM = int16

Initial state = 0 RPM

Access Programs

setSpeedController(int d_Velocity)

Maintains vehicle RPM of rear wheel equal to d_Velocity as an average of the last five rotations

		o_CurrentRPM
Current_RPM_Average > (d_Velocity- Tolerance)	d_Velocity = 0	0
	d_Velocity ≠ 0	CurrentRPM - 1
Current_RPM_Average < (d_Velocity+ Tolerance)	d_Velocity = 0	0
	d_Velocity ≠ 0	CurrentRPM + 1
(d_Velocity- Tolerance) < Current_RPM_Average > (d_Velocity+ Tolerance)		No Change

Constructor

init (*Hall Effect Sensor Pin: int8*)
init (Tolerance: int 8)

7.1.2 MID

Implementation

Var
setSpeedController(int d_Velocity)

inputs: d_Velocity
Outputs: Motor.SetVelocity(RPM)
Updates: None

		o_CurrentRPM
getSpeed() > (d_Velocity- Tolerance)	d_Velocity = 0	MotorControlOut.setVelocity(0);
	d_Velocity ≠ 0	MotorControlOut.setVelocity(d_Velocity -1);
getSpeed() < (d_Velocity+ Tolerance)	d_Velocity = 0	MotorControlOut.setVelocity(0);
	d_Velocity ≠ 0	MotorControlOut.setVelocity(d_Velocity +1);
(d_Velocity- Tolerance) < getSpeed() > (d_Velocity+ Tolerance)		MotorControlOut.setVelocity(d_Velocity);

getSpeed()

inputs: None
Outputs: Current_RPM_Average
Updates: None

	Current_RPM_Average
[∑60/(Time-Time$_{-1}$)] from i=1 … 5	Current_RPM_Average

"i" represents the last 5 RPM readings
The time difference is the amount of time between pulses.

7.2 Module: Ultra-Sonic

7.3.1 MIS

Reads the value from the front ultra-sonic sonic sensor to determine the distance to any obstacles.

Interface

Uses

None

Type

i_DistanceFrontRangeFinder = int 16

Initial state = 0 cm.

Access Programs

getDistanceFront(i_DistanceFrontRangeFinder:int 16)

Returns the current distance to nearest obstacle in front of the system

	i_DistanceFrontRangeFinder
Polled	New Reading
Not Polled	No Change

Constructor

init (*Ultra Sonic Pin Number: int8*)

7.3.2 MID

Implementation

Var

uint16_t UltrasonicAVG()

Inputs: None
Outputs: i_DistanceFrontRangeFinder
Updates: None

	i_DistanceFrontRangeFinder
UltrasonicDist() = ***polled***	New Reading
UltrasonicDist() = ***not polled***	No Change

Int UltrasonicDist()
Inputs: m_Distance
Outputs: i_Distance
Updates: None

Pseudo code:

The Ultra-Sonic sends out a signal by sending out a high pulse of two or more seconds, with a low pulse before the start of the high pulse and a low pulse at the end of the high pulse, to ensure a clean high pulse:

```
pinMode(UltraPin, OUTPUT);
digitalWrite(UltraPin, LOW);
delayMicroseconds(2);
digitalWrite(UltraPin, HIGH);
delayMicroseconds(5);
digitalWrite(UltraPin, LOW);
```

In order to read the Ultra-Sonic, the Ultra-Sonic is set to input mode and the duration of the receiving high pulse, is the time from the sending of the ping till the reception of its echo off an object.

```
pinMode(UltraPin, INPUT);
duration = pulseIn(UltraPin, HIGH);
```

The speed of sound is 29 microseconds per centimeters, thus the distance to the object can be calculated as such:

(duration / 29 / 2);

7.3 Module: PathFollow

7.5.1 MIS

Responsible for directing the vehicle to follow the specified path provided by the PathCorrection module.

Interface

Uses

PathCorrection

iTX

Type

c_Velocity = Int16

No Change implies previous value, with the initial state being 0 m/s.

c_TurnAngle = Int16

No Change implies previous value, with the initial state being 0 degrees.

Access Programs

getVelocity: c_Velocity

Returns the current system speed.

getTurnAngle: c_TurnAngle

Returns the current turning angle of the system.

setVelocity

Determines the current speed of the vehicle.

	Vehicle Speed Status c_Velocity
Path_Choice.getDesiredPath = FORWARD	k_VelocityMax
Path_Choice.getDesiredPath = LEFT[x]	k_VelocityMax - 20*x
Path_Choice.getDesiredPath = RIGHT[x]	k_VelocityMax - 20*x
Path_Choice.getDesiredPath = STOP	0

setTurnAngle

Determines the current turning angle of the vehicle.

	Vehicle Turning Angle Status c_TurnAngle
Path_Choice.getDesiredPath = FORWARD	0
Path_Choice.getDesiredPath = LEFT[x]	-60 * x/10
Path_Choice.getDesiredPath = RIGHT[x]	60 * x/10
Path_Choice.getDesiredPath = STOP	0

7.5.2 MID

Implementation

Var

getVelocity: c_Velocity

Inputs: m_Velocity
Outputs: getVelocity: c_Velocity
Updates: None

getTurnAngle: c_TurnAngle

Inputs: c_TurnAngle
Outputs: getTurnAngle: c_TurnAngle
Updates: None

setVelocity

Const k_VelocityMax = 300.00 cm/s

Inputs: e_DesiredPath = Path_Choice.getDesiredPath : e_Path
Outputs: c_Velocity

Updates: None

	c_Velocity
e_DesiredPath = FORWARD	k_VelocityMax
e_DesiredPath = LEFT[x]	k_VelocityMax – 20.00*x
e_DesiredPath = RIGHT[x]	k_VelocityMax – 20.00*x
e_DesiredPath = STOP	0

setTurnAngle

Inputs: e_DesiredPath = Path_Choice.getDesiredPath : e_Path
Outputs: c_TurnAngle
Updates: None

	c_TurnAngle
e_DesiredPath = FORWARD	0
e_DesiredPath = LEFT[x]	-60.00 * x/10
e_DesiredPath = RIGHT[x]	60.00 * x/10
e_DesiredPath = STOP	0

7.4 Module: BackupObstacleDetection

7.6.2 MIS

Responsible for detecting any undetected obstacles, by independently analyzing information from the camera module and information received directly from the range finder.

Uses

Camera

RangeFinder

Type

e_DesiredPath = (NONE, STOP)

NONE implies that no change will be made to the current path.
STOP implies that the current path is set to stop and prevent motion of the vehicle.

Access Programs

getDesiredPath: e_DesiredPath
Returns any changes in the path from the backup obstacle detection.

setDesiredPath
Determines any changes in the path from the backup obstacle detection.

	Desired Path Status e_DesiredPath
Camera.getImpact = True	STOP
Camera.getObsDist < 5 cm	STOP
RangeFinder.getDistanceFront < 5 cm	STOP

7.6.2 MID

Implementation

Var e_DesiredPath: e_Path

getDesiredPath: e_DesiredPath

Inputs: e_DesiredPath
Outputs: *getDesiredPath: e_DesiredPath*
Updates: None

setDesiredPath

Inputs: m_ObsDist = Camera.getObsDist
d_DistanceUltraSonic = RangeFinder.getDistanceFront
m_Impact = Camera.getImpact
Outputs: *e_DesiredPath*
Updates: None

	e_DesiredPath
m_Impact = True	STOP
m_ObsDist < 5.00 cm	STOP
d_DistanceUltraSonic < 5.00 cm	STOP

7.5 Module: MotorControlOut

7.7.2 MIS

Responsible for controlling the output velocity depending on the current velocity and the desired velocity.

Interface

Uses
EncoderSpeed

Type
Int16

Access Programs
getVelocity(o_CurrentSpeed:real)

Returns the current value of the system's velocity in cm/s.

setVelocity(d_Velocity:Int16)

Determines the desired velocity of the system.

	PWM_Velocity
Encoder_Speed.getVelocity > d_Velocity	PWM_Velocity(-1) -Vstep_down
Encoder_Speed.getVelocity < d_Velocity	PWM_Velocity(-1) +Vstep_up
Encoder_Speed.getVelocity = d_Velocity	No Change

(-1) implies previous state of variable.

Initial PWM_Velocity = 0

7.7.2 MID

Implementation

Var PWM_Velocity: Int16

getVelocity(o_CurrentSpeed:real)

Inputs: o_CurrentSpeed: Int16
Outputs: getVelocity(o_CurrentSpeed:real)
Updates: None

getVelocity = o_CurrentSpeed

setVelocity(d_Velocity:Int16)

Inputs: c_velocity(Int16) = EncoderSpeed.getVelocity: o_CurrentSpeed:real (conversion from real units to PWM signal), d_Velocity(Int16)
Outputs: None
Updates: PWM_Velocity: Int16

	PWM_Velocity
c_Velocity > d_Velocity	PWM_Velocity(-1) -Vstep_down
c_Velocity < d_Velocity	PWM_Velocity(-1) +Vstep_up
c_Velocity = d_Velocity	No Change

initVelocity

Inputs: None
Outputs: PWM_Velocity:Int16
Updates: None

PWM_Velocity = 0 {Vehicle is not moving initially}

7.6 Module: SteeringServoCommand

7.8.1 MIS

Responsible for monitoring and controlling the servomotor's angle to accomplish the desired turning angle.

Interface

Uses
None

Type
d_TurnAngle = (0, 180)
Desired turn angle of the front wheels in units of degrees.

Access Programs

setTurnAngle(d_TurnAngle:real)

Set the current angle equal to d_TurnAngle in degrees.

	d_TurnAngle
d_TurnAngle != d_TurnAngle(-1)	d_TurnAngle
d_TurnAngle = d_TurnAngle(-1)	No Change

Initial d_TurnAngle = 90 degrees

7.8.2 MID

Implementation

Var d_TurnAngle: Int16

setTurnAngle(d_TurnAngle:real)

Inputs: None
Outputs: None
Updates: d_TurnAngle: real

	d_TurnAngle
d_TurnAngle != d_TurnAngle(-1)	d_TurnAngle
d_TurnAngle = d_TurnAngle(-1)	No Change

initTurnAngle

Inputs: None
Outputs: d_TurnAngle:real
Updates: None

d_TurnAngle = 90 {vehicle is going straight and not turning}

7.7 Module: ImageCapture

7.9.1 MIS

This module captures the image from the camera and presents it in the format required. Note that new images are retrieved in a non-blocking fashion.

Uses

V4L2 (external dependency)

Types

t_OutputImage - uint8[][] ← Black and white image

Access Programs

WaitForImage()

Causes the current thread to stall until a processed image is ready. If a new image is not yet retrieved, this function returns instantly.

CheckForNewImage() : Bool

Checks if the current image is not yet processed. Returns true if it is not yet processed.

GetImage() : x_OutputImage

Retrieves the current image data. Note that this function can pause for extended periods while data is received.

7.9.2 MID

Private Types:

t_InputImage - uint8[][][] <- YUYV image data

Variables

m_hasNewImage: atboolean = false
m_InputImage: t_InputImage

Public Access Programs

WaitForImage()
Inputs: m_hasNewImage: boolean
Outputs: None
Updates: None

while(!m_hasNewImage);

CheckForNewImage() : boolean
Inputs: m_hasNewImage: boolean
Outputs: CheckForNewImage: boolean
Updates None

CheckForNewImage = m_hasNewImage

GetImage() : t_OutputImage
Inputs: m_InputImage: t_InputImage
Outputs: GetImage: t_OutputImage
Updates: m_hasNewImage: boolean

while(!m_hasNewImage)
m_hasNewImage = false
GetImage = convertImage(m_InputImage)

Private Access Programs

ConvertImage(input: t_InputImage): t_OutputImage
Inputs: input: t_InputImage
Outputs: ConvertImage: t_OutputImage
Updates: None

foreach(pixel in input/ConvertImage)

Pixel	ConvertImage
<128	0
>= 128	1

FetchImageThread()

Inputs: m_capture: Camera

Outputs: m_InputImage: t_InputImage, m_hasNewImage: boolean

while(1)

m_InputImage = m_caputre

m_hasNewImage = true

7.8 Module: ImageProcessing

7.10.1 MIS

Uses

ImageCapture

LocationDataCollector

Types

s_LaneInformation: ← Information for the current lane.
- m_ObstacleCheck1: Bool
- m_ObstacleCheck2: Bool
- m_ObstacleCheck3: Bool
- m_ObsDist1 : Int16 []
- m_ObsDist2 : Int16 []
- m_ObsDist3 : Int16 []
- m_LanePointsX : position []
- m_CurrentLane: Int8
- m_DistanceToCenter: vector

Access Programs

GetDefaultLane() : Int8

Returns the default lane that the vehicle started in. If unknown, this will return 0.

RefreshData()

Refreshes the data and pushes it to the data collector. Note this will block until completed.

Constructor(LocationDataCollector collector)

Constructs this image processor using the given data collector.

7.10.2 MID

Types

s_PotentialLine: <- Information on the line
m_CurRight: Point
m_CurLeft: Point
m_LastPoint: Point
m_Points: Vector<Point>

Variables

Camera: ImageCapture
DataCollector: LocationDataCollector
DefaultLane: int8

Public Access Programs

GetDefaultLane() : int8
Inputs: DefaultLane: int8
Outputs: GetDefaultLane: int8
Updates: None

GetDefaultLane = DefaultLane

m_ObstacleCheck1: boolean
m_ObstacleCheck2: boolean
m_ObstacleCheck3: boolean
m_ObsDist1 : int16 []
m_ObsDist2 : int16 []
m_ObsDist3 : int16 []
m_LanePointsX : position []
m_CurrentLane: int8
m_DistanceToCenter: vector

RefreshData()
Inputs: Camera: ImageCapture
Outputs: DataCollector: LocationDataCollector
Updates: None

Image = ImageCollector.GetImage()

PLines Vector<PotentialLine>
CurPLines: Vecor<PotentialLine>
foreach Line in Image
pixels = FindWhitePixels
foreach(pixel in pixels)

```
                    if(pixel is Close to Point in a PotentialLine in CurPLines)
                        Advance CurPLines
                    else
                        Add new Line
                foreach(untouched line in CurPLines)
                    Remove line from CurPLines

        Lines[4] = LongestLines in Plines
        Lines[3] = Midpoints between lines
        if(DefaultLane is 0)
            DefaultLane = find Closest to middle point

        CenterCurLane = (find Closest to middle point).point
        mDistanceToCenter = CenterCurLane - CenterPixelPoint

        Blobs[] = FindBlobs

        foreach(Blob in Blobs)
            Line = Closest Line to Blob
            m_ObstacleCheck[Line[Id]] = true
            m_ObstDist[Line[Id]]

    Constructor(LocationDataCollector collector)
        Inputs: collector: LocationDataCollector
        Outputs: LocalCollector:  LocationDataCollector
        Updates: None

        DataCollector = collector
```

7.9 Module: LocationDataCollector

7.11.1 MIS

Uses

ImageProcessing

Types

None

Access Programs

SetLocationData(ImageProcessing::s_LaneInformation: LaneInformation)

Sets the newly acquired data from the camera.

SetEncoderData(Int16: EncoderData)

Sets the encoder data from the sensors.

GetLocationData(): s_LaneInformation

Returns the latest lane information

GetEncoderData(): Int16

Returns the latest encoder data.

7.11.2 MID

Variables

LocalLaneInformation: ImageProcessing::s_LaneInformation
LocalEncoderData: int16

Public Access Programs

SetLocationData(ImageProcessing::s_LaneInformation: LaneInformation)
Inputs: LaneInformation: ImageProcessing::s_LaneInformation
Outputs: LocalLaneInformation: ImageProcessing::s_LaneInformation
Updates: None

LocalLaneInformation = LaneInformation

SetEncoderData(int16: EncoderData)
Inputs: EncoderData: int16
Outputs: LocalEncoderData: in16

LocalEncoderData = EncoderData

GetLocationData(): s_LaneInformation
Inputs: LocalLocationData: ImageProcessing::s_LaneInformation
Outputs: GetLocationData: ImageProcessing::s_LaneInformation
Updates: None

GetLocationData = LocalationData

GetEncoderData(): int16
Inputs: LocalEncoderData: int16
Outputs: GetEncoderData: int16

GetEncoderData = LocalEncoderData

7.10 Module: PathChoice

7.12.1 MIS

The PathChoice module is responsible for determining whether the system must deviate from its current path depending on the oncoming conditions of the track.

Interface

Uses

Camera

Type

o_NewLane = Int16
No change implies previous value.

e_CurrLane = Int16

Access Programs

o_Newlane
The new lane the system must change to.

Below is assuming both e_CurrLane and e_DefaultLane are 1

m_ObsCheck[X]	m_ObsDis1[1]	m_ObsDist2[1]	m_ObsDist3[1]	o_NewLane
[0,0,0]	>50	>50	>50	1
[1,0,0]	[20,50]	>50	>50	2
[1,0,0]	<20	>50	>50	2
[1,1,0]	[20,50]	[20,50]	>50	2
[1,1,0]	[20,50]	<20	>50	1
[1,1,0]	<20	[20,50]	>50	2
[1,1,0]	<20	<20	>50	1
[1,1,1]	[20,50]	[20,50]	[20,50]	1

[1,1,1]	<20	[20,50]	[20,50]	2
[1,1,1]	<20	<20	[20,50]	1
[1,1,1]	<20	<20	<20	1
[1,1,1]	[20,50]	[20,50]	<20	1
[1,1,1]	[20,50]	<20	<20	1
[1,1,1]	[20,50]	<20	[20,50]	1
[1,1,1]	<20	[20,50]	<20	2
[0,1,1]	>50	[20,50]	[20,50]	1
[0,1,1]	>50	<20	[20,50]	1
[0,1,1]	>50	<20	<20	1
[0,1,1]	>50	[20;50]	<20	1
[0,0,1]	>50	>50	[20,50]	1
[0,0,1]	>50	>50	<20	1
[1,0,1]	[20,50]	>50	[20,50]	2
[1,0,1]	<20	>50	[20,50]	2
[1,0,1]	<20	>50	<20	2
[1,0,1]	[20,50]	>50	<20	2
[0,1,0]	>50	[20,50]	>50	1
[0,1,0]	>50	<20	>50	1

7.12.2 MID

Implementation

Const k_distanceL = >50

k_distanceM = [20,50]

k_distanceS = <20

pickNewLane: o_Newlane

Inputs: m_ObsCheckX, m_ObsDistX[n], e_CurrLane, e_DefaultLane
Outputs: o_NewLane
Updates: e_CurrLane

*The implementation will consist of a series of pre-coded situations; this module will simply map the given inputs to the output that it corresponds too.

7.11 Module: PathCorrection

7.13.1 MIS

The purpose of this component is to correct the path of the system within a lane so that the system is travelling in the middle of the lane for the majority of the time.

Uses

Camera

PathChoice

Type

o_DesiredPath = e_Path

Access Programs

o_DesiredPath
The next state of motion the system shall take.

Condition	m_ObsDistX[n]	Distance to lane center	o_DesiredPath
o_NewLane - e_CurrLane = 0	>20	m_DistancetoCenter>0	RIGHT[m_DistancetoCenter+1]
	<20	DON'T CARE	STOP
	>20	m_DistancetoCenter<0	LEFT[abs(m_DistancetoCenter) +1]
o_NewLane - e_CurrLane = 1	DON'T CARE	DON'T CARE	RIGHT[8]
o_NewLane - e_CurrLane = -1	DON'T CARE	DON'T CARE	LEFT[8]

7.13.2 MID

Const k_AdjLane = 5 deg
k_ChgLane = 15 deg

Var i_Norm = abs(m_DistancetoCenter / 5)

makeDesiredPath: o_DesiredPath

Input: m_DistancetoCenter, e_CurrLane, o_NewLane, m_ObsDistX[n]
Output: m_DesiredPath
Update: i_Norm

*This function will take information from its inputs and adjust the position of the vehicle in proportion to the deviation from the center of the track. In the case that is moves to another lane, it will move with a fixed angle on the wheels until it reaches the middle of the lane. At which point it will resume correcting the path for the center of the lane.

7.12 Module: EStop

7.14.1 MIS

The purpose of this component is to come to an emergency stop in the event there is a critical failure within the system.

Uses

PathFollow

BackupObstacleDetection

Type

c_Velocity = Int16

No Change implies previous value, with the initial state being 0 m/s.

c_TurnAngle = Int16

No Change implies previous value, with the initial state being 0 degrees.

Access Programs

sendDataEStop(c_Estop)

This data will either be PathFollow data or in the event an emergency situation an emergency stop signal will be sent to oTX module.

7.14.2 MID

Implementation

Var

sendDataEStop()

Inputs: getVelocity(c_velocity:int 16)
Outputs: *sendDataEStop()*
Updates: None

	c_Estop
PathFollow.getVelocity == 0	Estop
PathFollow.getVelocity <> 0	PathFollow.getVelocity

7.13 Module: iTX (ARDUINO → BB-XM)

7.15.1 MIS

Uses

RangeFinder

Type

REF 5.2

Access Programs

GetRFData()

Pulls data from Range Finder module.

sendSerial(GetRFData()) (packet described in 5.2)

Transfers data byte by byte to the BB-xM.

readSerial()

Receives data from the BB-xM.

7.15.2 MID

Implementation

Var

Implementation

Var

sendSerial(GetRFData()):

Inputs: o_DistanceFrontRangeFinder:Int16
Outputs: Packet 5.2
Updates: None

readSerial:

Inputs: Packet 5.2
Outputs: d_Velocity, d_TurnAngle
Updates: None

GetRFData:

Inputs: None
Outputs: : o_DistanceFrontRangeFinder:Int16
Updates: None

7.14 Module: oTX (BB-XM→ ARDUINO)

7.16.1 MIS

Uses

EStop

Type

REF 5.1

Access Programs

txData(packet described in 5.2)

Transfers data byte by byte to the Arduino.

RxEStopData()

Receives data pushed by the EStop module.

7.16.2 MID

Implementation

Var

txData:

Inputs: c_Velocity: Int16, cTurnAngle:Int16
Outputs: Packet 5.1
Updates: None

RxEStopData:

Inputs: c_Velocity: Int16, cTurnAngle:Int16
Outputs: c_Velocity: Int16, cTurnAngle:Int16
Updates: None

7.15 Module: iRX (BB-XM)

7.17.1 MIS

Uses

iTX

Type

REF 5.2.

Access Programs

rxData(packet described in 5.2)

Receives data byte by byte from Arduino forming a packet.

SendData(packet described in 5.2)

Pushes the complete packet to data holder module.

7.17.2 MID

Implementation

Var

rxData:

Inputs: Packet 5.2
Outputs: o_DistanceFrontRangeFinder:Int16, o_Rear:Int16 , o_CurrentSpeed:Int16
Updates: None

SendData:

Inputs: o_DistanceFrontRangeFinder:Int16, o_Rear:Int16 , o_CurrentSpeed:Int16
Outputs: o_DistanceFrontRangeFinder:Int16, o_Rear:Int16 , o_CurrentSpeed:Int16
Updates: None

7.16 Module: oRX (ARDUINO)

7.18.1 MIS

Uses

oTX

Type

REF 5.1.

Access Programs

rxData(packet described in 5.1)

Receives data byte by byte from BB-xM forming a packet.

SendData(packet described in 5.1)

Pushes the complete packet to data holder module.

7.18.2 MID

Implementation

Var

rxData:

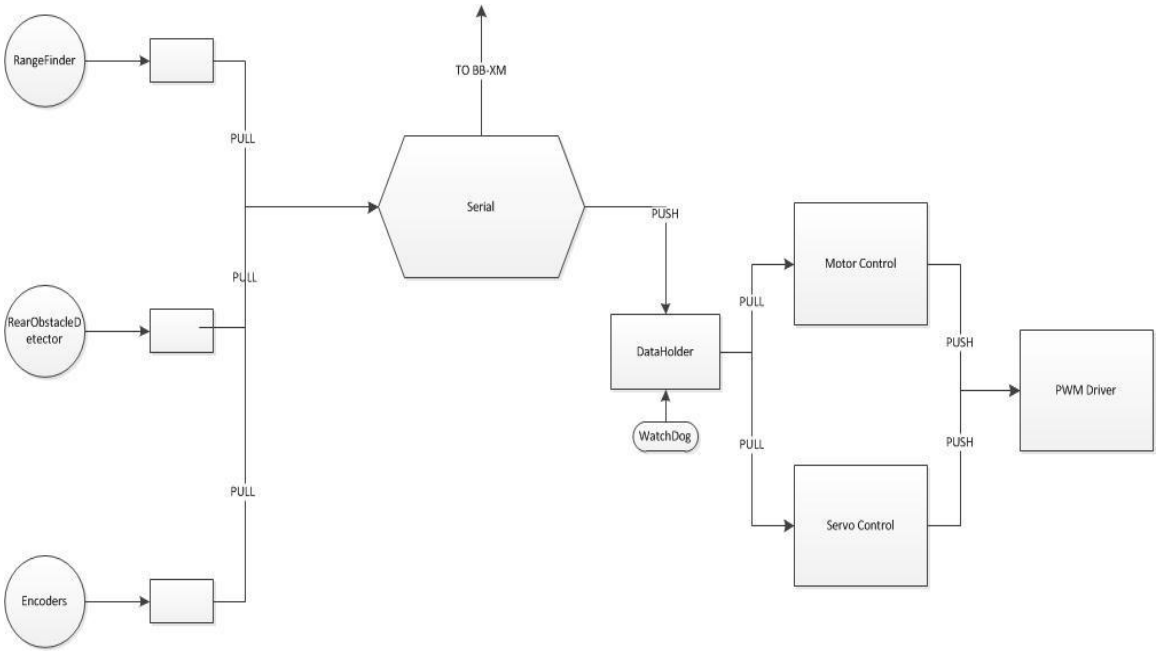
Inputs: Packet 5.1
Outputs: c_Velocity: Int16, cTurnAngle:Int16
Updates: None

SendData:

Inputs: c_Velocity: Int16, cTurnAngle:Int16
Outputs: c_Velocity: Int16, cTurnAngle:Int16
Updates: None

8. Data Transfer Model

8.1 ARDUINO



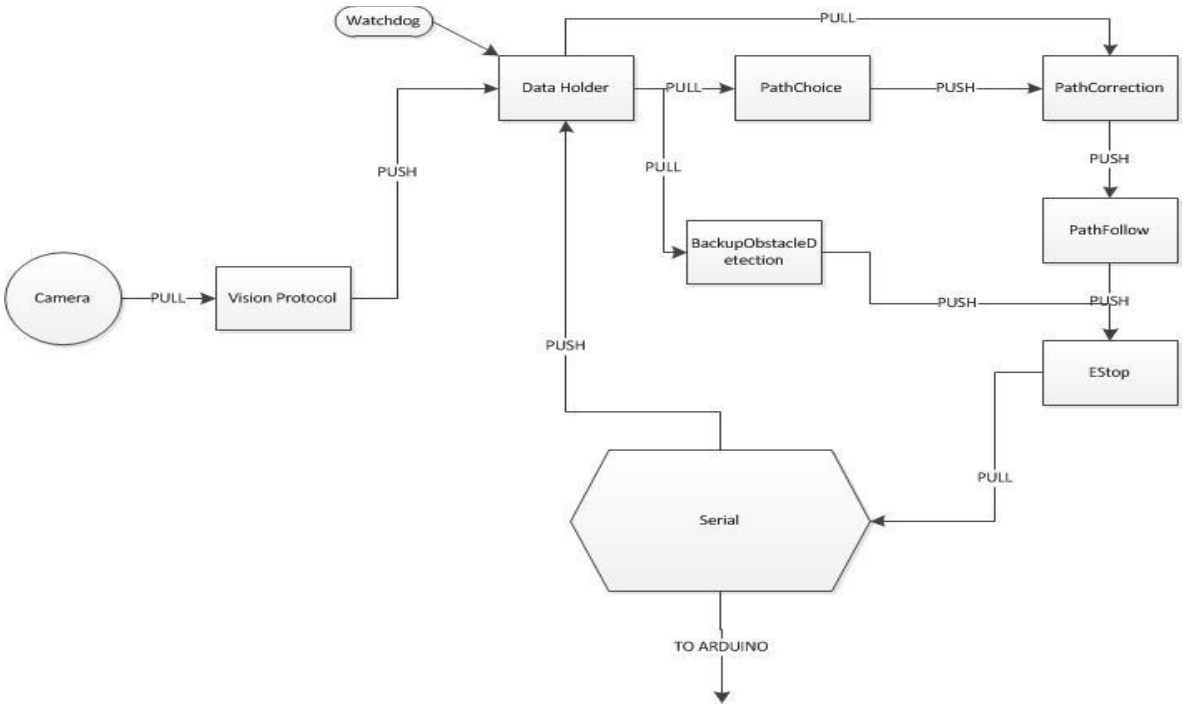
8.2 ARDUINO TIMING

Order of events in system:

1. Receive data from Beagleboard	2. Acknowledge and unpack package	3. Set motor and servo	4. Get data from front range finder	5. Get encoder data	6. Package information	7. Send data to the Beagleboard

The limiting factor with timing on the Arduino is the ultra-sonic sensor, which has a maximum response time of 18.5 ms. Unpacking, implementing new motor and servo control, and packing of sensor information is all measured in clock cycles and therefore adds no significant time to a frame. However, the other limiting factor is the transmission of data between the BeagleBoard and the Arduino. However, based on testing a 20 fps target frame is sufficient to meet all requirements. This corresponds to a frame size of 50 ms, and therefore will leave approximately 30 ms to send and receive data from the beagleboard. Lastly, the Hall Effect sensors are interrupt based and therefore timing is difficult to determine, however, through testing this has been shown to add no more than one ms to the frame.

8.3 BB-xM



8.4 BEAGLEBOARD TIMING

The limiting factor on the BeagleBoard is the time taken to process the camera image. Most other pieces, due to the nature of the system, can occur behind the scenes and will not disrupt the processing. If the camera can be processed in 30ms, then there should little to no issue.

APPENDIX A

1. Definitions, Acronyms and Abbreviations

Name	Definition
Object	Inanimate solid mass
Lose physical control	At any moment any one of the four tires loses contact with the ground
Lose electronic control	The vehicles control system loses power, browns out, or loses communication with any devices on the vehicle
Environment	The carpet area provided by client (REF SRS 2.4)
Road	The area contained within the thick white lines (white 2” duct tape) (REF SRS 2.4)
Line	White 3M hockey tape (REF SRS 2.4)
Desired Path	A route in which the vehicle will be able to travel unimpeded
Lane	The area between two lines
Obstacle	An object that is impeding the vehicles intended path or future intended path
Acceleration	To increase the rate of speed
Decelerate	To decrease the rate of speed
Vehicle/ System	The volume encompassing the body of the car, including chassis and any sensors
SRS	Software Requirement Specifications
Client	Dr. Alan Wassyng of McMaster University

2. VARIABLES MASTER LIST

2.1 Monitored Variables

SRS:

Monitored Name	Description	Type	Units
m_Environment	A signal monitoring the environment directly in front of the vehicle for detection of lanes at a frequency that allows the system to detect all three lanes k_LaneDistance (20 cm) ahead of the vehicle within a tolerance of +/- k_LaneDistaneTolerance (5 cm). (as per section 5.11.1)	Digital	cm
m_Velocity	A signal monitoring the speed (m/s) of the system in relation to the environment.	Analog	m/s
m_IsCrossedLane	A signal monitoring whether the vehicle has crossed a line (white 3M hockey tape). Refer to Requirement 5.11.0001	Digital	boolean

m_SystemState	A variable that is triggered by an event that causes the system to be in an uncontrolled state. Refer to Requirement 5.11.00002	Digital	boolean
m_ObsCheckX (1 <= X <= 3)	Store whether an obstacle is present in lane x.	Digital	boolean
m_BoundaryLeft	The distance between the vehicle and the left boundary of the track with respect to the vehicle.	Digital	cm
m_BoundaryRight	The distance between the vehicle and the right boundary of the track with respect to the vehicle.	Digital	cm
m_ObsDist	The distance between the vehicle and an approaching obstacle in the current lane.	Digital	cm
m_DistEdge	The distance between the vehicle and the edge of the adjacent lane	Digital	cm
m_ForwardBool	The variable used as a check to make sure that that the system is moving in a forward direction and never in the reverse direction.	Digital	boolean
m_Impact	Determine if system has collided with another object in the environment	Digital	boolean
m_LaneBool	Check if system is within boundaries of lane or not	Digital	boolean

DSD:

Monitored Name	Variable Type	Description	Type	Units
m_cameraImage	N/A	A signal monitoring the environment directly in front of the vehicle for detection of lanes at a frequency that allows the system to detect all three lanes k_LaneDistance (20 cm) ahead of the vehicle within a tolerance of +/- k_LaneDistaneTolerance (5 cm).	Digital	N/A
m_CurrentSpeedLeft	Binary	A signal monitoring rotations of the left rear wheel of the system	Digital	cm/s
m_CurrentSpeedRight	Binary	A signal monitoring rotations of the right rear wheel of the system	Digital	cm/s
m_IsCrossedLane	Bool	A signal monitoring whether the vehicle has crossed a line (white 3M hockey tape).	Digital	boolean
m_SystemState	Bool	A variable that is triggered by an event that causes the system to be in an uncontrolled state.	Digital	boolean

m_ObsCheckX[n] X = [0,3] n = 1, 2, 3, ...	Bool	Store whether an obstacle is present in lane x.	Digital	boolean
m_BoundaryLeft	Int16	The distance between the vehicle and the left boundary of the track with respect to the vehicle.	Digital	cm
m_BoundaryRight	Int16	The distance between the vehicle and the right boundary of the track with respect to the vehicle.	Digital	cm
m_ObsDist	Int16	The distance between the vehicle and an approaching obstacle in the current lane.	Digital	cm
m_DistEdgeLeft	Int16	The distance between the vehicle and the edge of the adjacent lane to the left	Digital	cm
m_DistEdgeRight	Int16	The distance between the vehicle and the edge of the adjacent lane to the right	Digital	cm
m_ForwardBool	Bool	The variable used as a check to make sure that that the system is moving in a forward direction and never in the reverse direction.	Digital	boolean
m_DistancetoCenter	Int16	The distance between the center of the center of the lane and the midpoint between two lanes.	Digital	cm
m_Impact	Bool	Determine if system has collided with another object in the environment	Digital	boolean
m_LaneBool	Bool	Check if system is within boundaries of lane or not	Digital	boolean

2.2 Controlled Variables

Control Name	Variable Type	Description	Type	Units
c_Velocity	Int16	The controlled variable, the system speed	Analog (PWM)	m/s
c_TurnAngle	Int16	The turning angle of the wheels of the system	Analog (PWM)	Degrees

2.3 Enumerated Variables

Enumerated name	Variable Type	Description	Type

y_CurrLane	Int16	The current lane the vehicle is situated in	Digital
y_DefaultLane	Int16	The lane the vehicle began the track in, also known as e_CurrLane0	Digital
y_Path	{FORWARD, LEFT, RIGHT, STOP}	An enumerated type used for changing the travelling direction of the vehicle.	Digital
y_DesiredPath	{e_Path, Int16}	An enumerated type used for adding magnitude to the LEFT and RIGHT travelling directions in e_Path	Digital

3. CONSTANTS MASTER LIST

Constant	Variable Type	Value	Units	Description
k_LaneDistaneTolerance	Int16	5	cm	Acceptable tolerance when measuring distance between lanes
k_StopMin	Int16	10	cm	Minimum stopping distance
k_StopMax	Int16	20	cm	Maximum stopping distance when vehicle detects a need to stop
k_DecelerationMax	Int16	-0.02	m/s^2	Maximum rate of negative acceleration
k_DecelerationMin	Int16	0	m/s^2	Minimum rate of negative acceleration
k_AccelerationMax	Int16	10	m/s^2	Maximum rate of positive acceleration
k_ObsDetAcc	Int16	0.1	s	The accuracy needed to detect an object approaching the current lane.
k_RoadWidth	Int16	100	cm	The width of the road.
k_StopTolerance	Int16	5	cm	Acceptable tolerance when measuring stopping distance.
k_Stop	Int16	0	m/s	The stop state when the system is not in motion.
k_StopDistanceRatio	Int16	TBD	TBD	The stop distance determined by the current speed of the vehicle as soon as stop mechanism is activated.
k_ObsMaxDist	Int16	20	cm	The maximum distance where the system should detect an obstacle.
k_ObsMinDist	Int16	10	cm	The minimum distance where the system should detect an obstacle.
k_LapTimeMax	Int16	2	minute(s)	This is the max time for 1 lap of the track

k_LaneWidth	Int16	30	cm	The width of one lane.
k_ObsConfidence	Int16	2	s	Confidence in which the system can react in the allocated that an obstacle is present in the current lane
k_BatteryLife	Int16	1	hour	Minimum required battery life.
k_BatteryDrain	Int16	500	mAh	Rate at which battery loses charge
k_Circumference	Int16	x	cm	Circumference of the wheels
k_PointsDistance	Integer	1	cm	Distance between points describing lines

4. Tables

4.1 Path Choice Table

e_DefaultLane	e_CurrentLane	m_ObsCheck[X]	m_ObsDist[1]	m_ObsDist2[1]	m_ObsDist3[1]	o_NewLane
1	1	[0,0,0]	>50	>50	>50	1
1	1	[1,0,0]	[20,50]	>50	>50	2
1	1	[1,0,0]	<20	>50	>50	2
1	1	[1,1,0]	[20,50]	[20,50]	>50	2
1	1	[1,1,0]	[20,50]	<20	>50	1
1	1	[1,1,0]	<20	[20,50]	>50	2
1	1	[1,1,0]	<20	<20	>50	1
1	1	[1,1,1]	[20,50]	[20,50]	[20,50]	1
1	1	[1,1,1]	<20	[20,50]	[20,50]	2
1	1	[1,1,1]	<20	<20	[20,50]	1
1	1	[1,1,1]	<20	<20	<20	1
1	1	[1,1,1]	[20,50]	[20,50]	<20	1
1	1	[1,1,1]	[20,50]	<20	<20	1
1	1	[1,1,1]	[20,50]	<20	[20,50]	1
1	1	[1,1,1]	<20	[20,50]	<20	2
1	1	[0,1,1]	>50	[20,50]	[20,50]	1
1	1	[0,1,1]	>50	<20	[20,50]	1
1	1	[0,1,1]	>50	<20	<20	1
1	1	[0,1,1]	>50	[20,50]	<20	1
1	1	[0,0,1]	>50	>50	[20,50]	1
1	1	[0,0,1]	>50	>50	<20	1
1	1	[1,0,1]	[20,50]	>50	[20,50]	2
1	1	[1,0,1]	<20	>50	[20,50]	2
1	1	[1,0,1]	<20	>50	<20	2

1	**1**	[1,0,1]	[20,50]	>50	<20	2
1	**1**	[0,1,0]	>50	[20,50]	>50	1
1	**1**	[0,1,0]	>50	<20	>50	1
1	**2**	[0,0,0]	>50	>50	>50	1
1	**2**	[1,0,0]	[20,50]	>50	>50	2
1	**2**	[1,0,0]	<20	>50	>50	2
1	**2**	[1,1,0]	[20,50]	[20,50]	>50	3
1	**2**	[1,1,0]	[20,50]	<20	>50	3
1	**2**	[1,1,0]	<20	[20,50]	>50	3
1	**2**	[1,1,0]	<20	<20	>50	3
1	**2**	[1,1,1]	[20,50]	[20,50]	[20,50]	1
1	**2**	[1,1,1]	<20	[20,50]	[20,50]	2
1	**2**	[1,1,1]	<20	<20	[20,50]	3
1	**2**	[1,1,1]	<20	<20	<20	2
1	**2**	[1,1,1]	[20,50]	[20,50]	<20	1
1	**2**	[1,1,1]	[20,50]	<20	<20	1
1	**2**	[1,1,1]	[20,50]	<20	[20,50]	1
1	**2**	[1,1,1]	<20	[20,50]	<20	2
1	**2**	[0,1,1]	>50	[20,50]	[20,50]	1
1	**2**	[0,1,1]	>50	<20	[20,50]	1
1	**2**	[0,1,1]	>50	<20	<20	1
1	**2**	[0,1,1]	>50	[20,50]	<20	1
1	**2**	[0,0,1]	>50	>50	[20,50]	1
1	**2**	[0,0,1]	>50	>50	<20	1
1	**2**	[1,0,1]	[20,50]	>50	[20,50]	2
1	**2**	[1,0,1]	<20	>50	[20,50]	2
1	**2**	[1,0,1]	<20	>50	<20	2
1	**2**	[1,0,1]	[20,50]	>50	<20	2
1	**2**	[0,1,0]	>50	[20,50]	>50	1
1	**2**	[0,1,0]	>50	<20	>50	1
1	**3**	[0,0,0]	>50	>50	>50	2
1	**3**	[1,0,0]	[20,50]	>50	>50	2
1	**3**	[1,0,0]	<20	>50	>50	2
1	**3**	[1,1,0]	[20,50]	[20,50]	>50	3
1	**3**	[1,1,0]	[20,50]	<20	>50	3
1	**3**	[1,1,0]	<20	[20,50]	>50	3
1	**3**	[1,1,0]	<20	<20	>50	3
1	**3**	[1,1,1]	[20,50]	[20,50]	[20,50]	2
1	**3**	[1,1,1]	<20	[20,50]	[20,50]	2

1	**3**	[1,1,1]	<20	<20	[20,50]	3
1	**3**	[1,1,1]	<20	<20	<20	3
1	**3**	[1,1,1]	[20,50]	[20,50]	<20	2
1	**3**	[1,1,1]	[20,50]	<20	<20	3
1	**3**	[1,1,1]	[20,50]	<20	[20,50]	3
1	**3**	[1,1,1]	<20	[20,50]	<20	2
1	**3**	[0,1,1]	>50	[20,50]	[20,50]	2
1	**3**	[0,1,1]	>50	<20	[20,50]	3
1	**3**	[0,1,1]	>50	<20	<20	3
1	**3**	[0,1,1]	>50	[20,50]	<20	2
1	**3**	[0,0,1]	>50	>50	[20,50]	2
1	**3**	[0,0,1]	>50	>50	<20	2
1	**3**	[1,0,1]	[20,50]	>50	[20,50]	2
1	**3**	[1,0,1]	<20	>50	[20,50]	2
1	**3**	[1,0,1]	<20	>50	<20	2
1	**3**	[1,0,1]	[20,50]	>50	<20	2
1	**3**	[0,1,0]	>50	[20,50]	>50	3
1	**3**	[0,1,0]	>50	<20	>50	3
2	1	[0,0,0]	>50	>50	>50	2
2	1	[1,0,0]	[20,50]	>50	>50	2
2	1	[1,0,0]	<20	>50	>50	2
2	1	[1,1,0]	[20,50]	[20,50]	>50	2
2	1	[1,1,0]	[20,50]	<20	>50	1
2	1	[1,1,0]	<20	[20,50]	>50	2
2	1	[1,1,0]	<20	<20	>50	1
2	1	[1,1,1]	[20,50]	[20,50]	[20,50]	2
2	1	[1,1,1]	<20	[20,50]	[20,50]	2
2	1	[1,1,1]	<20	<20	[20,50]	1
2	1	[1,1,1]	<20	<20	<20	1
2	1	[1,1,1]	[20,50]	[20,50]	<20	2
2	1	[1,1,1]	[20,50]	<20	<20	1
2	1	[1,1,1]	[20,50]	<20	[20,50]	1
2	1	[1,1,1]	<20	[20,50]	<20	2
2	1	[0,1,1]	>50	[20,50]	[20,50]	1
2	1	[0,1,1]	>50	<20	[20,50]	1
2	1	[0,1,1]	>50	<20	<20	1
2	1	[0,1,1]	>50	[20,50]	<20	1
2	1	[0,0,1]	>50	>50	[20,50]	2
2	1	[0,0,1]	>50	>50	<20	2
2	1	[1,0,1]	[20,50]	>50	[20,50]	2

2	1	[1,0,1]	<20	>50	[20,50]	2
2	1	[1,0,1]	<20	>50	<20	2
2	1	[1,0,1]	[20,50]	>50	<20	2
2	1	[0,1,0]	>50	[20,50]	>50	1
2	1	[0,1,0]	>50	<20	>50	1
2	**2**	[0,0,0]	>50	>50	>50	2
2	**2**	[1,0,0]	[20,50]	>50	>50	2
2	**2**	[1,0,0]	<20	>50	>50	2
2	**2**	[1,1,0]	[20,50]	[20,50]	>50	3
2	**2**	[1,1,0]	[20,50]	<20	>50	3
2	**2**	[1,1,0]	<20	[20,50]	>50	3
2	**2**	[1,1,0]	<20	<20	>50	3
2	**2**	[1,1,1]	[20,50]	[20,50]	[20,50]	2
2	**2**	[1,1,1]	<20	[20,50]	[20,50]	2
2	**2**	[1,1,1]	<20	<20	[20,50]	3
2	**2**	[1,1,1]	<20	<20	<20	2
2	**2**	[1,1,1]	[20,50]	[20,50]	<20	2
2	**2**	[1,1,1]	[20,50]	<20	<20	1
2	**2**	[1,1,1]	[20,50]	<20	[20,50]	3
2	**2**	[1,1,1]	<20	[20,50]	<20	2
2	**2**	[0,1,1]	>50	[20,50]	[20,50]	1
2	**2**	[0,1,1]	>50	<20	[20,50]	1
2	**2**	[0,1,1]	>50	<20	<20	1
2	**2**	[0,1,1]	>50	[20,50]	<20	1
2	**2**	[0,0,1]	>50	>50	[20,50]	2
2	**2**	[0,0,1]	>50	>50	<20	2
2	**2**	[1,0,1]	[20,50]	>50	[20,50]	2
2	**2**	[1,0,1]	<20	>50	[20,50]	2
2	**2**	[1,0,1]	<20	>50	<20	2
2	**2**	[1,0,1]	[20,50]	>50	<20	2
2	**2**	[0,1,0]	>50	[20,50]	>50	3
2	**2**	[0,1,0]	>50	<20	>50	3
2	**3**	[0,0,0]	>50	>50	>50	2
2	**3**	[1,0,0]	[20,50]	>50	>50	2
2	**3**	[1,0,0]	<20	>50	>50	2
2	**3**	[1,1,0]	[20,50]	[20,50]	>50	3
2	**3**	[1,1,0]	[20,50]	<20	>50	3
2	**3**	[1,1,0]	<20	[20,50]	>50	3
2	**3**	[1,1,0]	<20	<20	>50	3

2	**3**	[1,1,1]	[20,50]	[20,50]	[20,50]	2
2	**3**	[1,1,1]	<20	[20,50]	[20,50]	2
2	**3**	[1,1,1]	<20	<20	[20,50]	3
2	**3**	[1,1,1]	<20	<20	<20	3
2	**3**	[1,1,1]	[20,50]	[20,50]	<20	2
2	**3**	[1,1,1]	[20,50]	<20	<20	3
2	**3**	[1,1,1]	[20,50]	<20	[20,50]	3
2	**3**	[1,1,1]	<20	[20,50]	<20	2
2	**3**	[0,1,1]	>50	[20,50]	[20,50]	2
2	**3**	[0,1,1]	>50	<20	[20,50]	3
2	**3**	[0,1,1]	>50	<20	<20	3
2	**3**	[0,1,1]	>50	[20,50]	<20	2
2	**3**	[0,0,1]	>50	>50	[20,50]	2
2	**3**	[0,0,1]	>50	>50	<20	2
2	**3**	[1,0,1]	[20,50]	>50	[20,50]	2
2	**3**	[1,0,1]	<20	>50	[20,50]	2
2	**3**	[1,0,1]	<20	>50	<20	2
2	**3**	[1,0,1]	[20,50]	>50	<20	2
2	**3**	[0,1,0]	>50	[20,50]	>50	3
2	**3**	[0,1,0]	>50	<20	>50	3
3	**1**	[0,0,0]	>50	>50	>50	2
3	**1**	[1,0,0]	[20,50]	>50	>50	2
3	**1**	[1,0,0]	<20	>50	>50	2
3	**1**	[1,1,0]	[20,50]	[20,50]	>50	2
3	**1**	[1,1,0]	[20,50]	<20	>50	1
3	**1**	[1,1,0]	<20	[20,50]	>50	2
3	**1**	[1,1,0]	<20	<20	>50	1
3	**1**	[1,1,1]	[20,50]	[20,50]	[20,50]	2
3	**1**	[1,1,1]	<20	[20,50]	[20,50]	2
3	**1**	[1,1,1]	<20	<20	[20,50]	1
3	**1**	[1,1,1]	<20	<20	<20	1
3	**1**	[1,1,1]	[20,50]	[20,50]	<20	2
3	**1**	[1,1,1]	[20,50]	<20	<20	1
3	**1**	[1,1,1]	[20,50]	<20	[20,50]	1
3	**1**	[1,1,1]	<20	[20,50]	<20	2
3	**1**	[0,1,1]	>50	[20,50]	[20,50]	1
3	**1**	[0,1,1]	>50	<20	[20,50]	1
3	**1**	[0,1,1]	>50	<20	<20	1
3	**1**	[0,1,1]	>50	[20,50]	<20	1
3	**1**	[0,0,1]	>50	>50	[20,50]	2

3	**1**	[0,0,1]	>50	>50	<20	2
3	**1**	[1,0,1]	[20,50]	>50	[20,50]	2
3	**1**	[1,0,1]	<20	>50	[20,50]	2
3	**1**	[1,0,1]	<20	>50	<20	2
3	**1**	[1,0,1]	[20,50]	>50	<20	2
3	**1**	[0,1,0]	>50	[20,50]	>50	1
3	**1**	[0,1,0]	>50	<20	>50	1
3	**2**	[0,0,0]	>50	>50	>50	3
3	**2**	[1,0,0]	[20,50]	>50	>50	3
3	**2**	[1,0,0]	<20	>50	>50	3
3	**2**	[1,1,0]	[20,50]	[20,50]	>50	3
3	**2**	[1,1,0]	[20,50]	<20	>50	3
3	**2**	[1,1,0]	<20	[20,50]	>50	3
3	**2**	[1,1,0]	<20	<20	>50	3
3	**2**	[1,1,1]	[20,50]	[20,50]	[20,50]	3
3	**2**	[1,1,1]	<20	[20,50]	[20,50]	3
3	**2**	[1,1,1]	<20	<20	[20,50]	3
3	**2**	[1,1,1]	<20	<20	<20	2
3	**2**	[1,1,1]	[20,50]	[20,50]	<20	2
3	**2**	[1,1,1]	[20,50]	<20	<20	1
3	**2**	[1,1,1]	[20,50]	<20	[20,50]	3
3	**2**	[1,1,1]	<20	[20,50]	<20	2
3	**2**	[0,1,1]	>50	[20,50]	[20,50]	3
3	**2**	[0,1,1]	>50	<20	[20,50]	1
3	**2**	[0,1,1]	>50	<20	<20	1
3	**2**	[0,1,1]	>50	[20,50]	<20	1
3	**2**	[0,0,1]	>50	>50	[20,50]	2
3	**2**	[0,0,1]	>50	>50	<20	2
3	**2**	[1,0,1]	[20,50]	>50	[20,50]	2
3	**2**	[1,0,1]	<20	>50	[20,50]	2
3	**2**	[1,0,1]	<20	>50	<20	2
3	**2**	[1,0,1]	[20,50]	>50	<20	2
3	**2**	[0,1,0]	>50	[20,50]	>50	3
3	**2**	[0,1,0]	>50	<20	>50	3
3	**3**	[0,0,0]	>50	>50	>50	3
3	**3**	[1,0,0]	[20,50]	>50	>50	3
3	**3**	[1,0,0]	<20	>50	>50	3
3	**3**	[1,1,0]	[20,50]	[20,50]	>50	3
3	**3**	[1,1,0]	[20,50]	<20	>50	3

3	**3**	[1,1,0]	<20	[20,50]	>50	3
3	**3**	[1,1,0]	<20	<20	>50	3
3	**3**	[1,1,1]	[20,50]	[20,50]	[20,50]	3
3	**3**	[1,1,1]	<20	[20,50]	[20,50]	3
3	**3**	[1,1,1]	<20	<20	[20,50]	3
3	**3**	[1,1,1]	<20	<20	<20	3
3	**3**	[1,1,1]	[20,50]	[20,50]	<20	2
3	**3**	[1,1,1]	[20,50]	<20	<20	3
3	**3**	[1,1,1]	[20,50]	<20	[20,50]	3
3	**3**	[1,1,1]	<20	[20,50]	<20	2
3	**3**	[0,1,1]	>50	[20,50]	[20,50]	3
3	**3**	[0,1,1]	>50	<20	[20,50]	3
3	**3**	[0,1,1]	>50	<20	<20	3
3	**3**	[0,1,1]	>50	[20,50]	<20	2
3	**3**	[0,0,1]	>50	>50	[20,50]	2
3	**3**	[0,0,1]	>50	>50	<20	2
3	**3**	[1,0,1]	[20,50]	>50	[20,50]	2
3	**3**	[1,0,1]	<20	>50	[20,50]	2
3	**3**	[1,0,1]	<20	>50	<20	2
3	**3**	[1,0,1]	[20,50]	>50	<20	2
3	**3**	[0,1,0]	>50	[20,50]	>50	3
3	**3**	[0,1,0]	>50	<20	>50	3

5. SIMULATIONS

The purpose of the simulations was to get a good idea about the logic behind machine vision and test different methods to accomplish the goals of the project: lane detection and obstacle avoidance.

The simulations were primarily done on MATLAB but a specialized computer vision program called Roborealm was used as well. MATLAB provided the ability to perform simulations on the lane following algorithms while avoiding any details in the actual implementation of the algorithms. There were two main aspects of the simulations, the first was to verify the motion of the car by simulation the servo control and the second was to simulate lane detection on actual images of the track to determine if the midpoints were being properly calculated.

5.1 TURNING SERVO CONTROL (MATLAB)

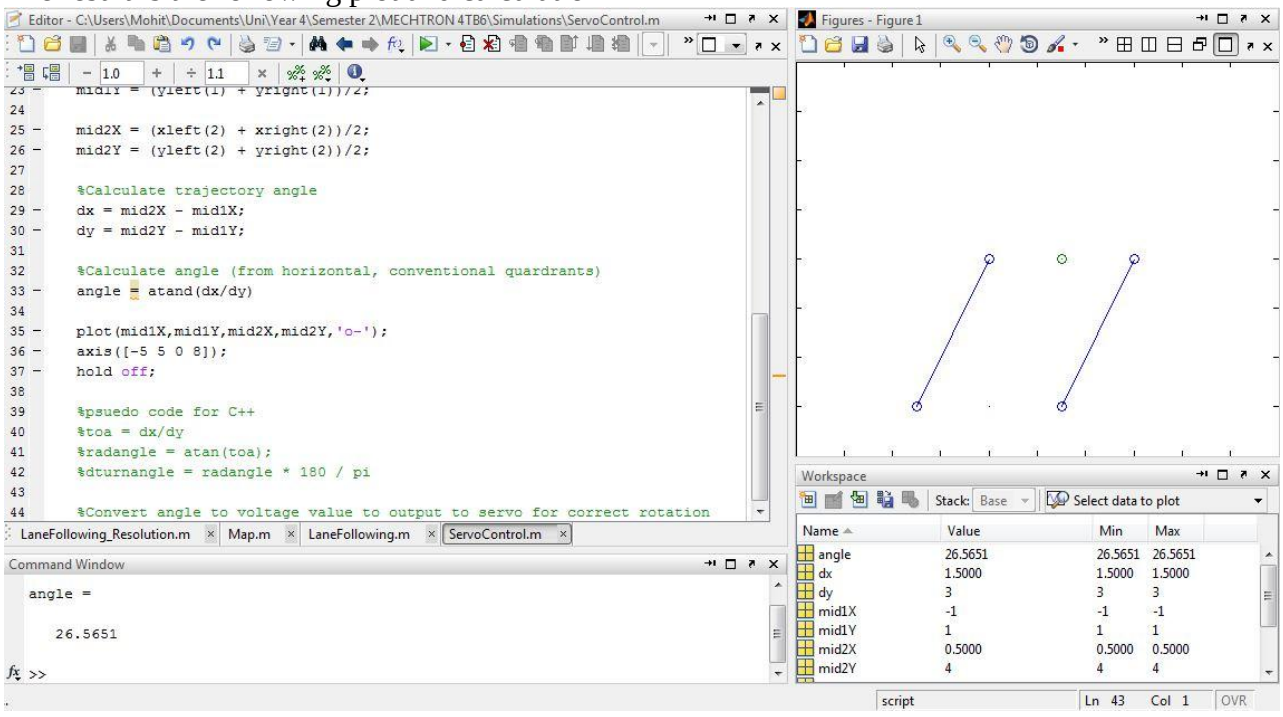
The first simulation done was a basic plot to calculate the angle the servo must turn to complete a segment of a curve.

The code:

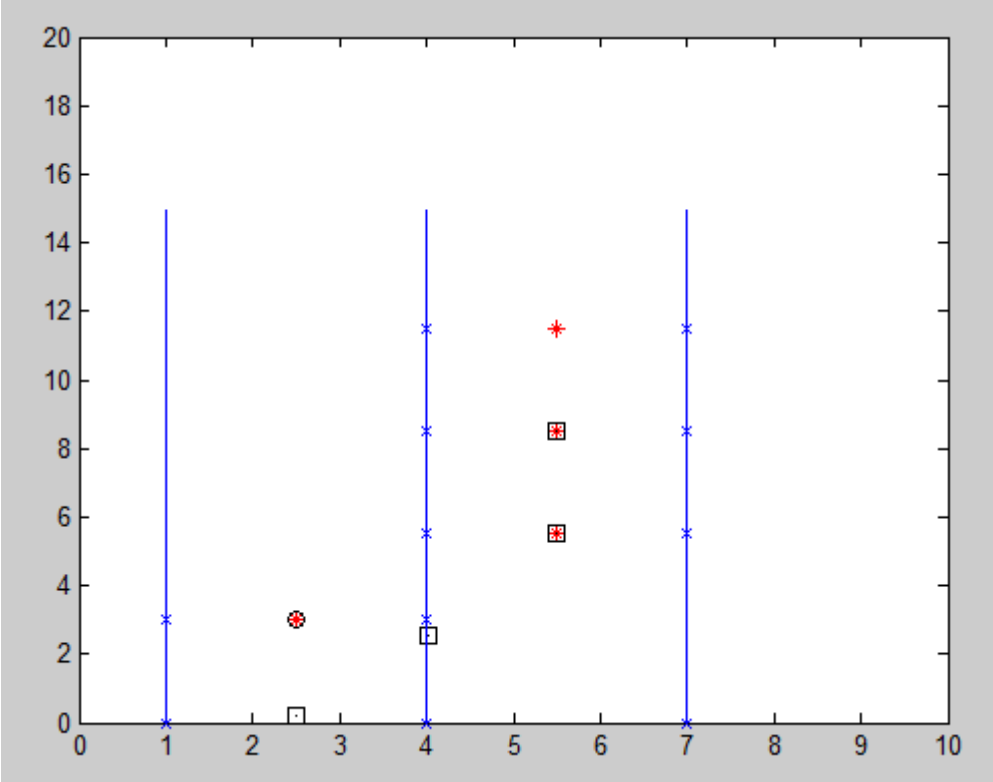
```
clc;
clear;

%Simulation for Turning

%Left boundary
%Note: each interval = 10 cm
xleft = [-2.5 -1];
yleft = [1 4];

plot(xleft,yleft,'o-');
hold on;

%Right boundary
xright = xleft + [3 3]; %Lanes are 30 cm wide
yright = yleft;%[1 6];

plot(xright,yright,'o-');
hold on;

%Calculate midpoints
mid1X = (xleft(1) + xright(1))/2;
mid1Y = (yleft(1) + yright(1))/2;
```

```
mid2X = (xleft(2) + xright(2))/2;
mid2Y = (yleft(2) + yright(2))/2;

%Calculate trajectory angle
dx = mid2X - mid1X;
dy = mid2Y - mid1Y;

%Calculate angle (from horizontal, conventional quardrants)
angle = atand(dx/dy)

plot(mid1X,mid1Y,mid2X,mid2Y,'o-');
axis([-5 5 0 8]);
hold off;

%psuedo code for C++
%toa = dx/dy
%radangle = atan(toa);
%dturnangle = radangle * 180 / pi

%Convert angle to voltage value to output to servo for correct rotation
```

The result is the following plot and calculation



The results show us that the servo control is directing the vehicle towards the midpoints so can stay within a lane. When this is performed with multiple segments we see vehicles path remains within the lane boundaries.



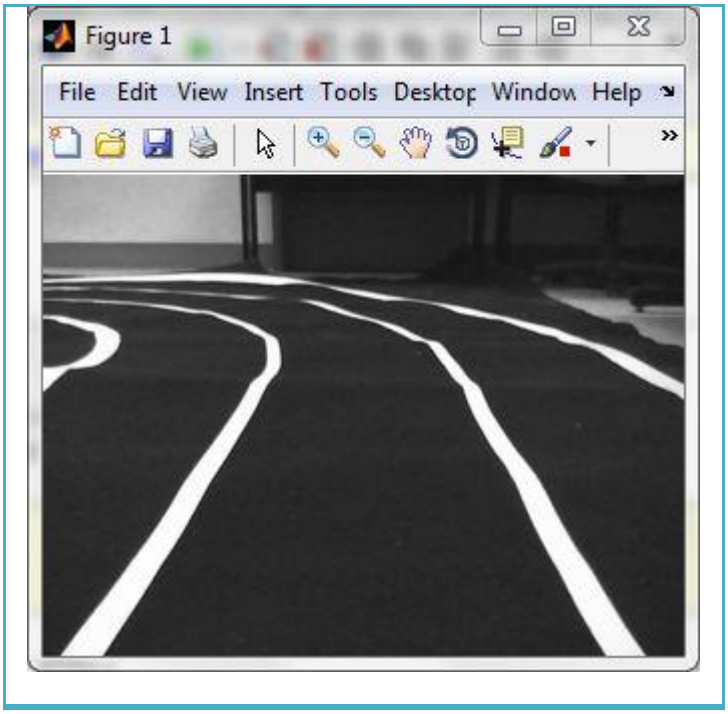
The circle is an obstacle and it was avoided.

5.2 LANE DETECTION (MATLAB)

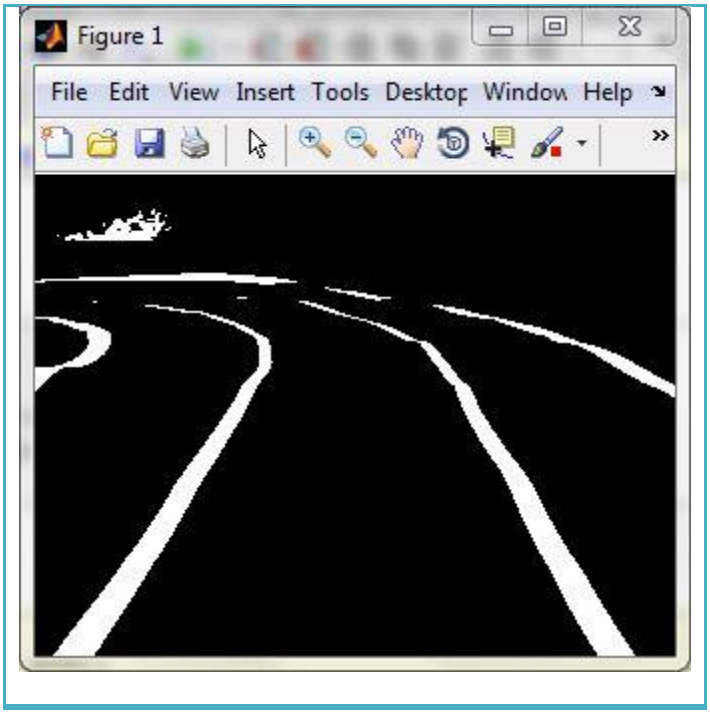
Using a spare webcam (not being used for the final design) the team took pictures of the track to analyze using MATLAB for lane detection.

The code along with resulting figures of each cell:

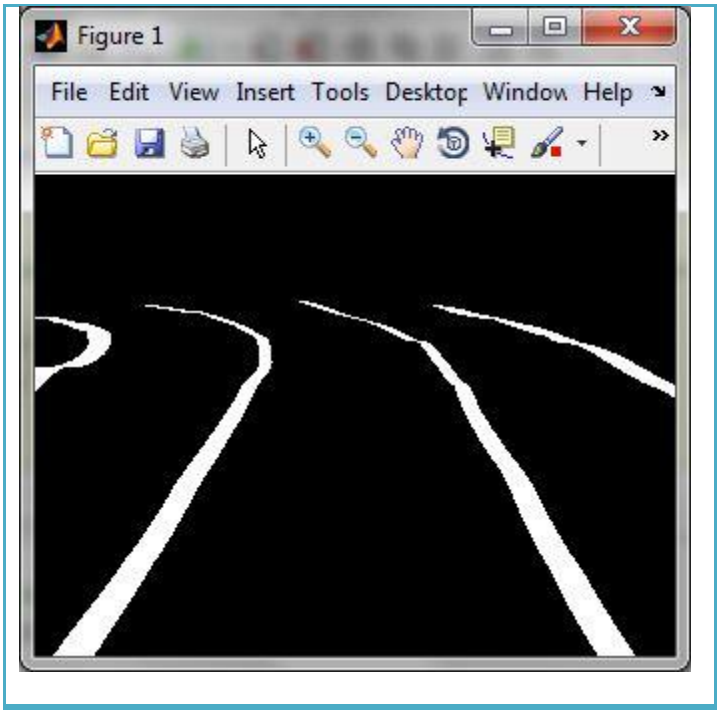
```
clear;
clc;
clf;
%%Lane Following
%%
%%Load Image
track = imread('track1.jpg');
track = imresize(track, [240 320]);
```



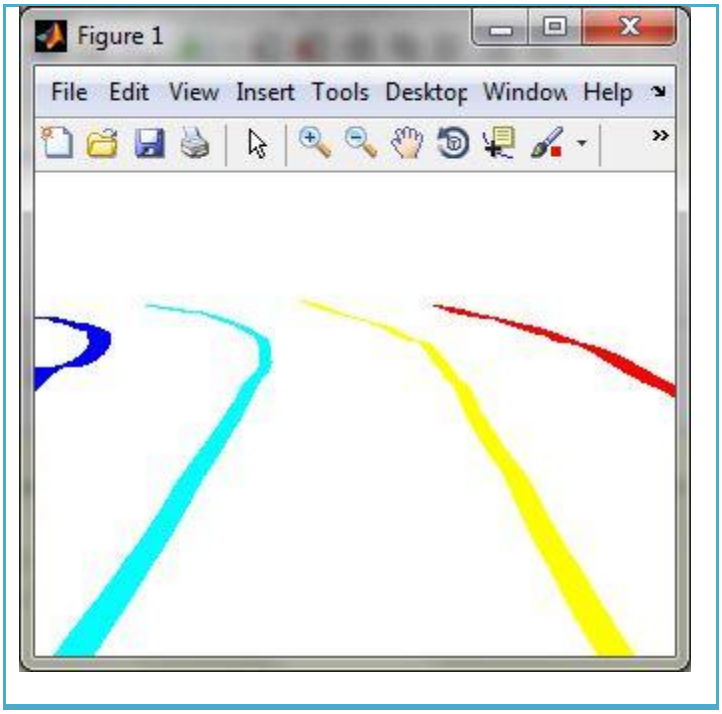

```
imshow(track);
%% Segment by Threshold
bwtrack = im2bw(track,.75);
```




```
imshow(bwtrack);

%% Clean image
lanes = bwareaopen(bwtrack, 370);
imshow(lanes);
```

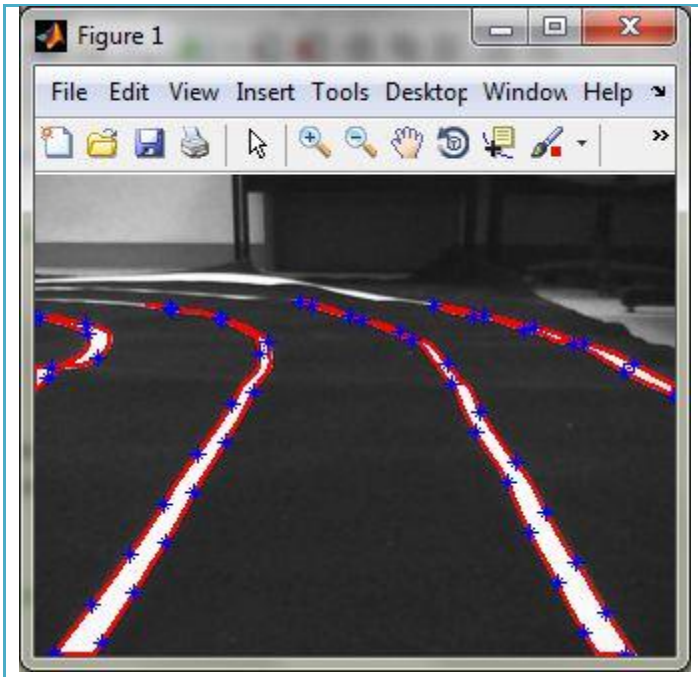


```
%% Find lanes
%L = store pixels of image as [0,1,2..] where 0 is no object and 1 is lane
%1 and so one
%B = array of pixel boundary locations
[B, L] = bwboundaries(lanes,'noholes');
numregions = max(L(:));
imshow(label2rgb(L));
```




```
%% Connected regions
stats = regionprops(L,'all');
%% Eccentricity
% Round object = 0, linear object = 1
shapes = [stats.Eccentricity];
keepers = find(shapes>.8);
%Note: instead of eccentricity in actual implementation we must use
%intensity difference to detect edges around the lanes.
%% Overlay lines over original image with edges
imshow(track);
hold on;

%Create outline
for index=1:length(keepers)
```

```
    outline = B{keepers(index)};
    line(outline(:,2),outline(:,1),'Color','r','LineWidth',2);
end

%Along mark points along edges
edX = [];
edY = [];
for i = 1:length(keepers)
    for edges = length(B{keepers(i),1})
        for bounds = 1:25:edges
            edX = B{keepers(i),1}(bounds,2);
            edY = B{keepers(i),1}(bounds,1);
            plot(edX, edY,'*','Color','b');
            hold on;
        end
    end
end
hold off;
```

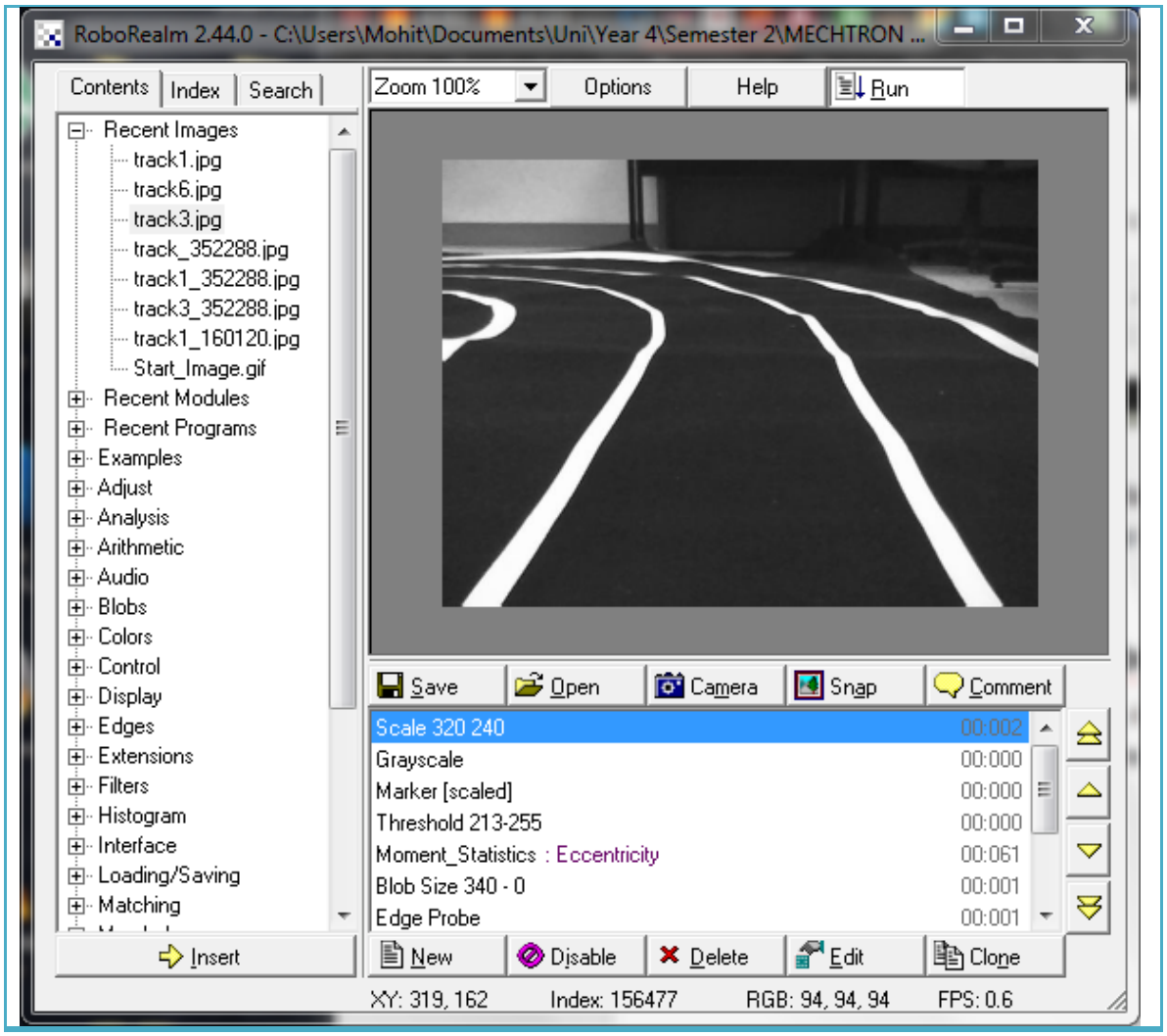


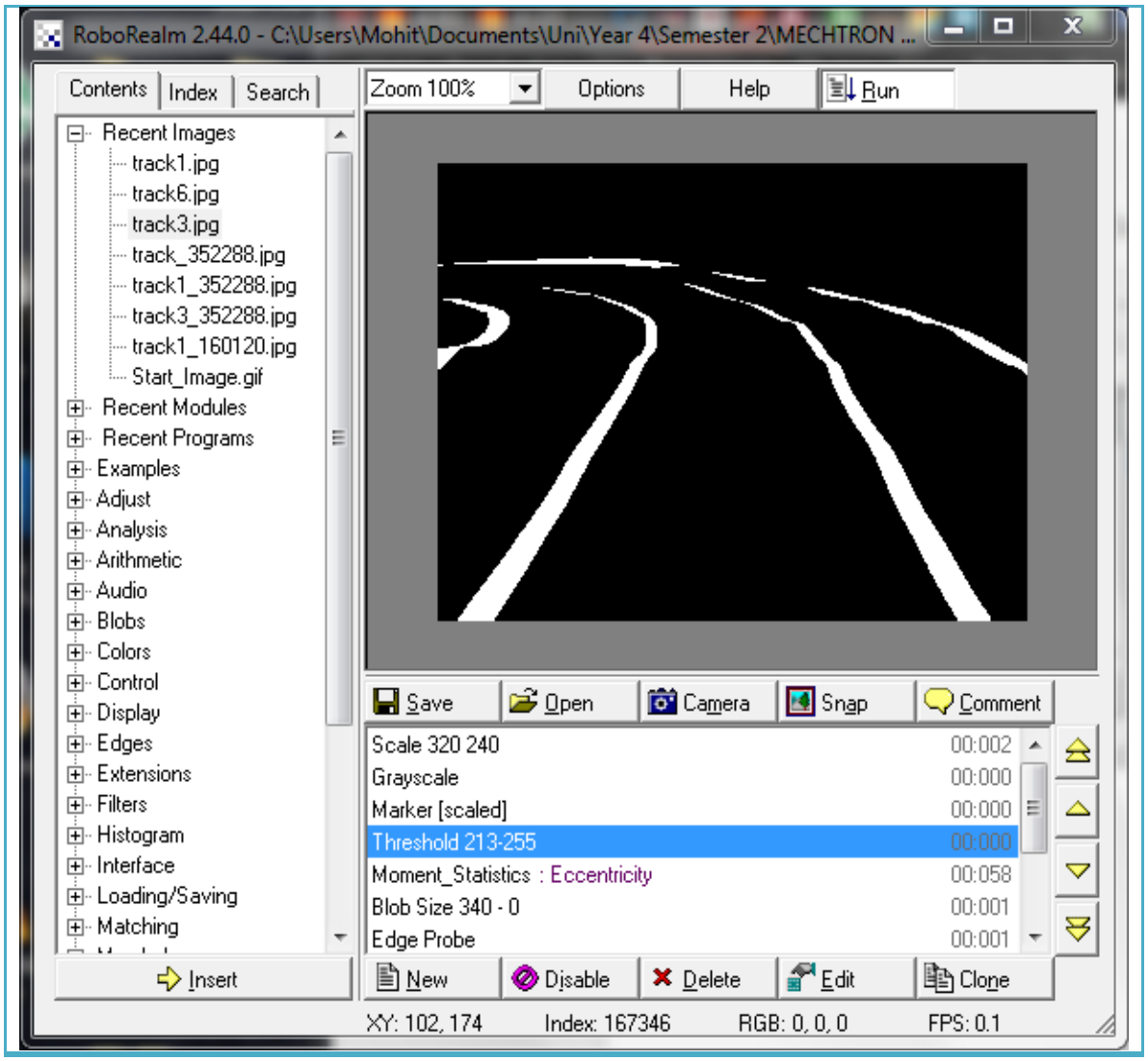
The result is the following image:

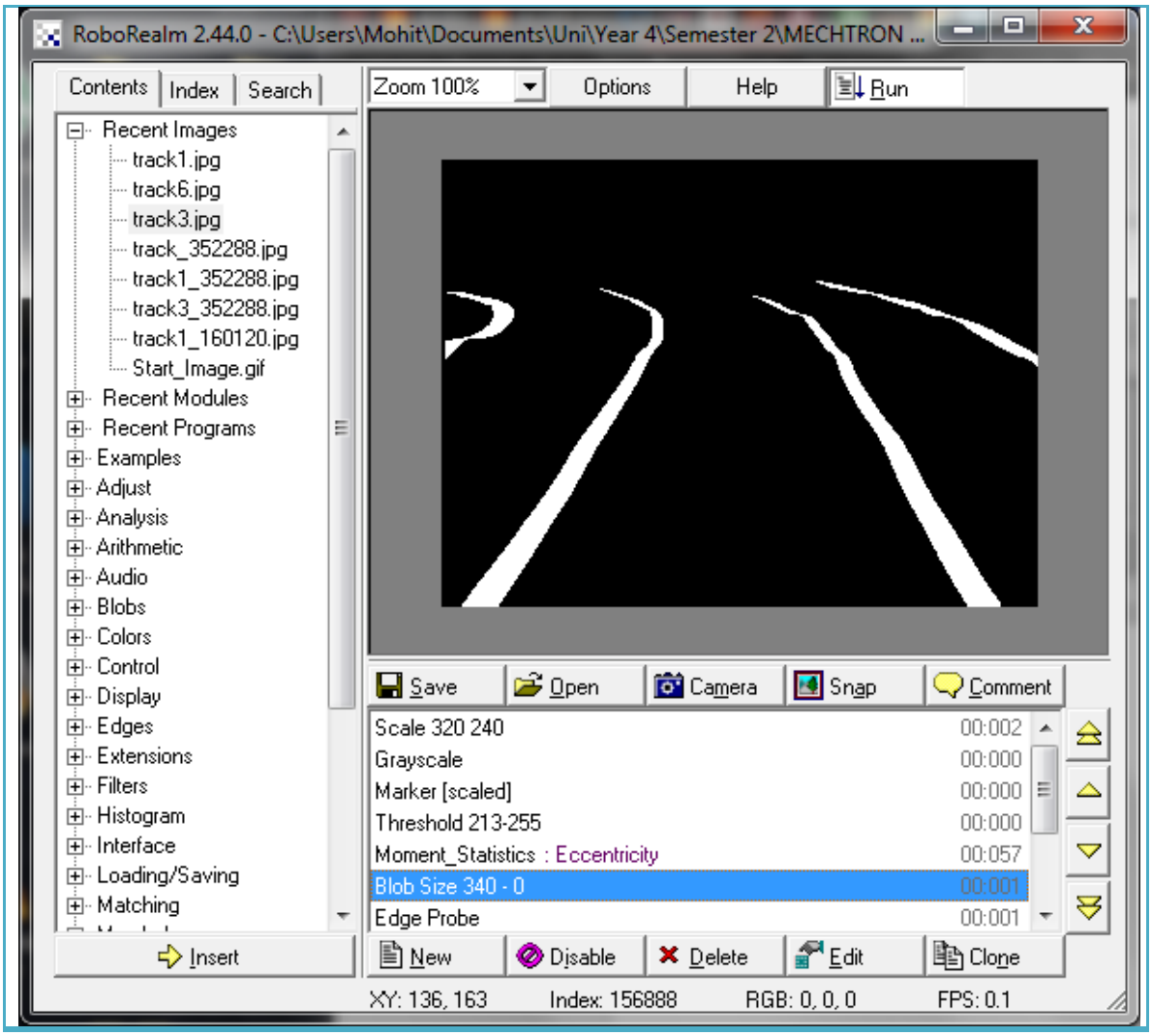
For later use, we can potentially connect the dots from one lane to adjacent lane and calculate the midpoint for the car to go to up ahead.

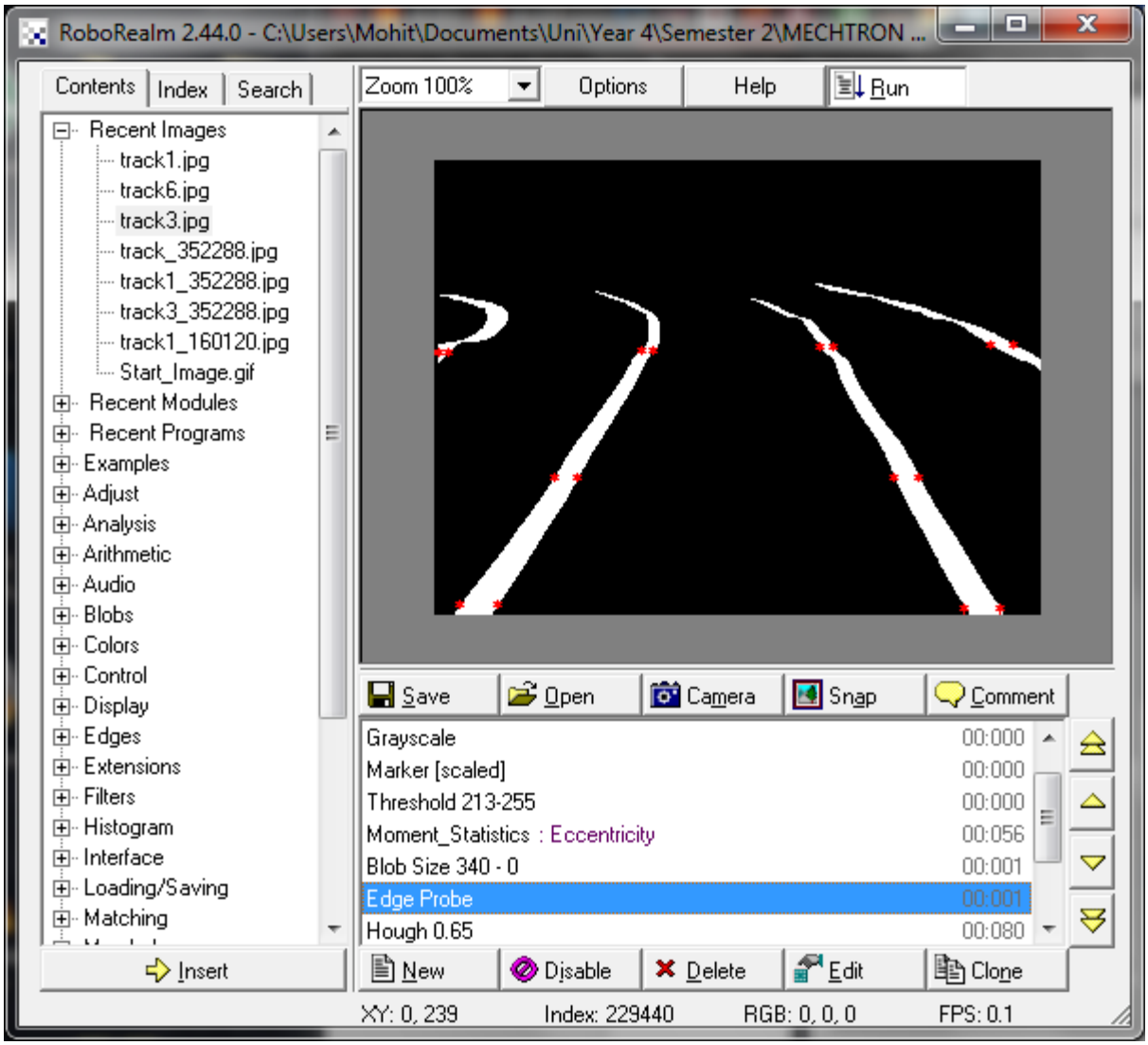
5.3 LANE DETECTION (ROBOREALM)

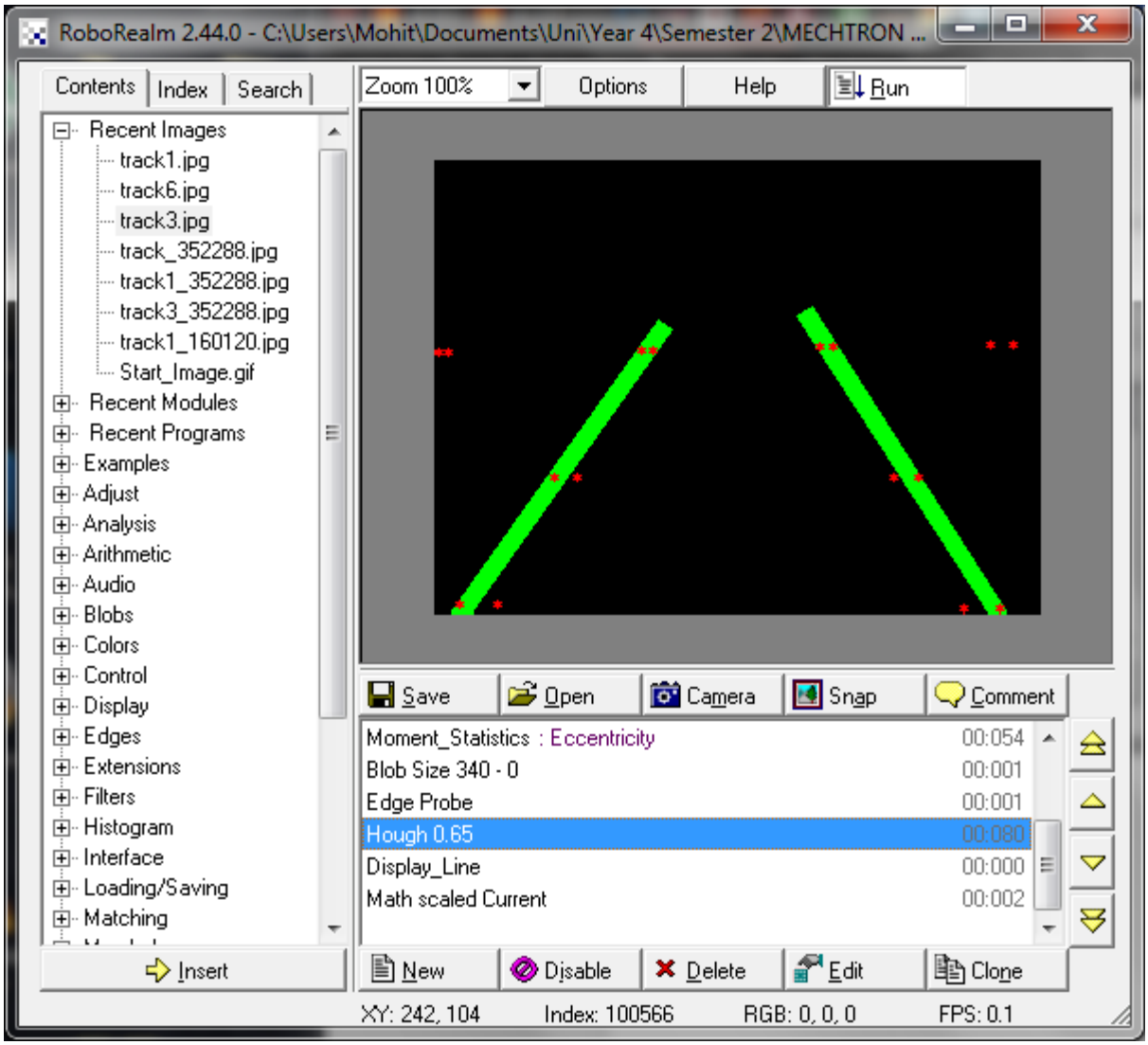
RoboRealm provides a very intuitive GUI to apply some of the same features that required hard coding in MATLAB. Using the same image:

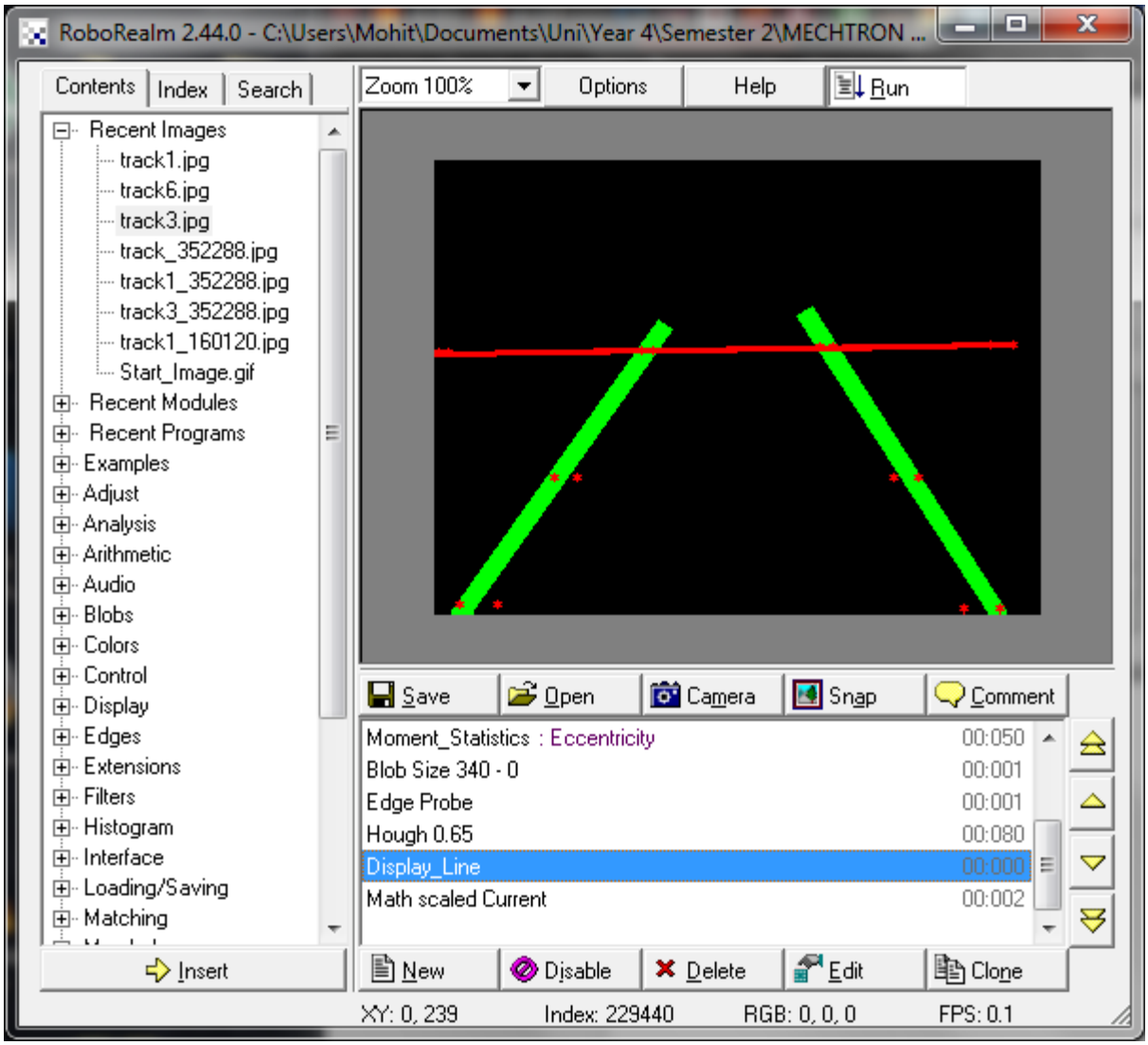


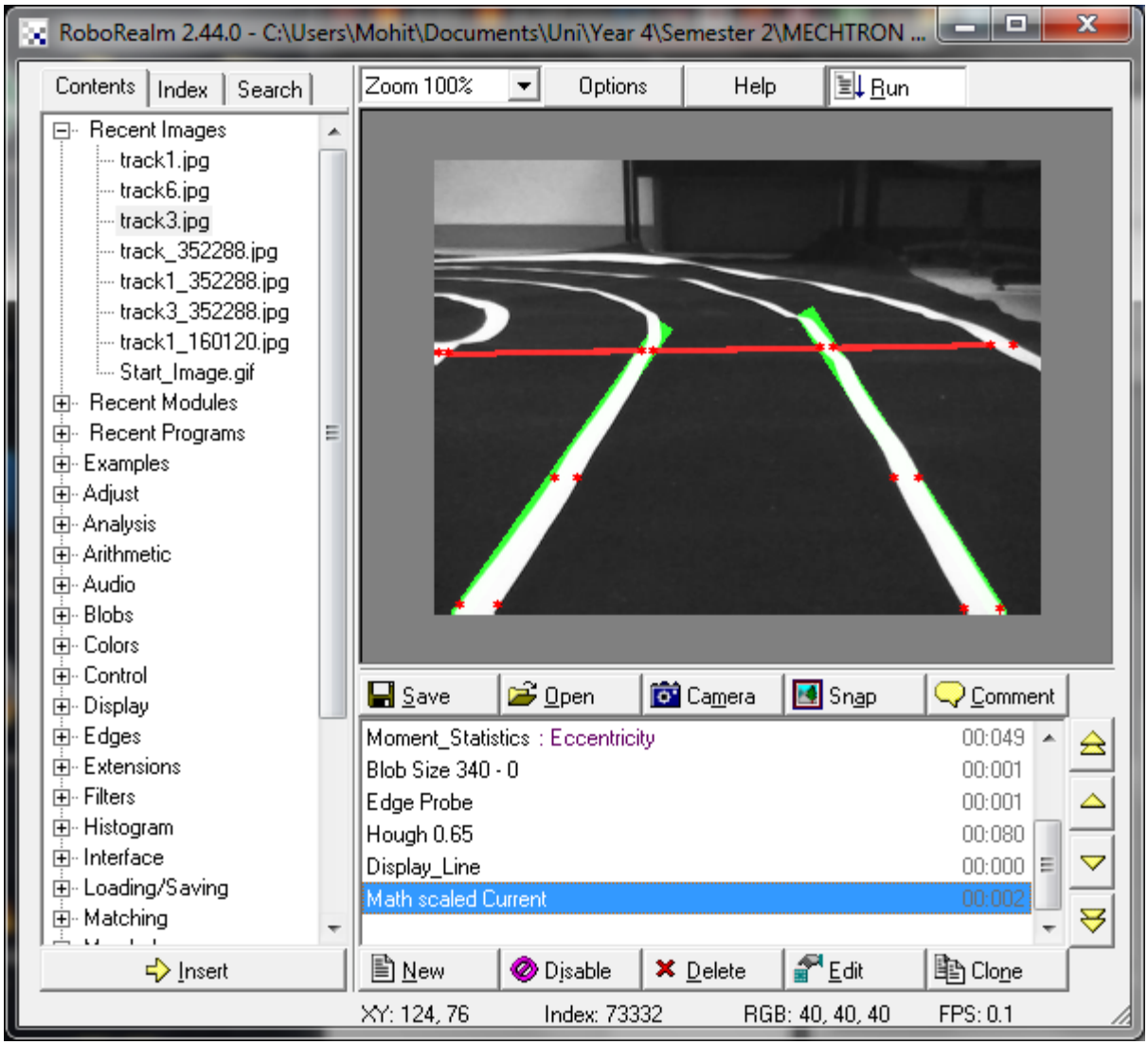

RoboRealm 2.44.0 - C:\Users\Mohit\Documents\Uni\Year 4\Semester 2\MECHTRON ...
Contents
Index
Search
Zoom 100%
Options
Help
Run
Recent Images
track1.jpg
track6.jpg
track3.jpg
track_352288.jpg
track1_352288.jpg
track3_352288.jpg
track1_160120.jpg
Start_Image.gif
Recent Modules
Recent Programs
Examples
Adjust
Analysis
Arithmetic
Audio
Blobs
Colors
Control
Display
Edges
Extensions
Filters
Histogram
Interface
Loading/Saving
Matching
Save
Open
Camera
Snap
Comment
Scale 320 240 00:002
Grayscale 00:000
Marker [scaled] 00:000
Threshold 213-255 00:000
Moment_Statistics : Eccentricity 00:058
Blob Size 340 - 0 00:001
Edge Probe 00:001
Insert
New
Disable
Delete
Edit
Clone
XY: 102, 174
Index: 167346
RGB: 0, 0, 0
FPS: 0.1


RoboRealm 2.44.0 - C:\Users\Mohit\Documents\Uni\Year 4\Semester 2\MECHTRON ...
Contents
Index
Search
Zoom 100%
Options
Help
Run
Recent Images
track1.jpg
track6.jpg
track3.jpg
track_352288.jpg
track1_352288.jpg
track3_352288.jpg
track1_160120.jpg
Start_Image.gif
Recent Modules
Recent Programs
Examples
Adjust
Analysis
Arithmetic
Audio
Blobs
Colors
Control
Display
Edges
Extensions
Filters
Histogram
Interface
Loading/Saving
Matching
Save
Open
Camera
Snap
Comment
Scale 320 240 00:002
Grayscale 00:000
Marker [scaled] 00:000
Threshold 213-255 00:000
Moment_Statistics : Eccentricity 00:057
Blob Size 340 - 0 00:001
Edge Probe 00:001
Insert
New
Disable
Delete
Edit
Clone
XY: 136, 163
Index: 156888
RGB: 0, 0, 0
FPS: 0.1

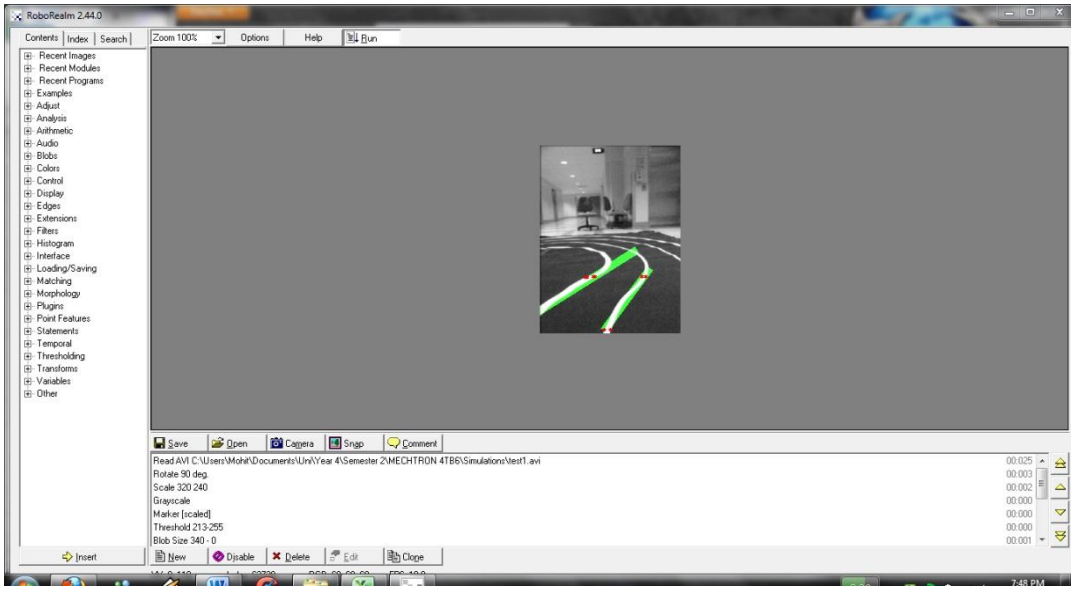

RoboRealm 2.44.0 - C:\Users\Mohit\Documents\Uni\Year 4\Semester 2\MECHTRON ...
Contents
Index
Search
Zoom 100%
Options
Help
Run
Recent Images
track1.jpg
track6.jpg
track3.jpg
track_352288.jpg
track1_352288.jpg
track3_352288.jpg
track1_160120.jpg
Start_Image.gif
Recent Modules
Recent Programs
Examples
Adjust
Analysis
Arithmetic
Audio
Blobs
Colors
Control
Display
Edges
Extensions
Filters
Histogram
Interface
Loading/Saving
Matching
Save
Open
Camera
Snap
Comment
Grayscale 00:000
Marker [scaled] 00:000
Threshold 213-255 00:000
Moment_Statistics : Eccentricity 00:056
Blob Size 340 - 0 00:001
Edge Probe 00:001
Hough 0.65 00:080
Insert
New
Disable
Delete
Edit
Clone
XY: 0, 239
Index: 229440
RGB: 0, 0, 0
FPS: 0.1


RoboRealm 2.44.0 - C:\Users\Mohit\Documents\Uni\Year 4\Semester 2\MECHTRON ...
Contents
Index
Search
Zoom 100%
Options
Help
Run
Recent Images
track1.jpg
track6.jpg
track3.jpg
track_352288.jpg
track1_352288.jpg
track3_352288.jpg
track1_160120.jpg
Start_Image.gif
Recent Modules
Recent Programs
Examples
Adjust
Analysis
Arithmetic
Audio
Blobs
Colors
Control
Display
Edges
Extensions
Filters
Histogram
Interface
Loading/Saving
Matching
Save
Open
Camera
Snap
Comment
Moment_Statistics : Eccentricity 00:054
Blob Size 340 - 0 00:001
Edge Probe 00:001
Hough 0.65 00:080
Display_Line 00:000
Math scaled Current 00:002
Insert
New
Disable
Delete
Edit
Clone
XY: 242, 104
Index: 100566
RGB: 0, 0, 0
FPS: 0.1


RoboRealm 2.44.0 - C:\Users\Mohit\Documents\Uni\Year 4\Semester 2\MECHTRON ...
Contents
Index
Search
Zoom 100%
Options
Help
Run
Recent Images
track1.jpg
track6.jpg
track3.jpg
track_352288.jpg
track1_352288.jpg
track3_352288.jpg
track1_160120.jpg
Start_Image.gif
Recent Modules
Recent Programs
Examples
Adjust
Analysis
Arithmetic
Audio
Blobs
Colors
Control
Display
Edges
Extensions
Filters
Histogram
Interface
Loading/Saving
Matching
Save
Open
Camera
Snap
Comment
Moment_Statistics : Eccentricity 00:050
Blob Size 340 - 0 00:001
Edge Probe 00:001
Hough 0.65 00:080
Display_Line 00:000
Math scaled Current 00:002
Insert
New
Disable
Delete
Edit
Clone
XY: 0, 239
Index: 229440
RGB: 0, 0, 0
FPS: 0.1



After using Roborealm on a still image, a test video was taken using the camera intended to be used and the results were the following.



This is not an accurate representation as with our current hardware mounts the camera was not able to be placed in its normal orientation. This gives a good idea of the types of filters needed to be coded and what is possible to do in video processing in terms of frames per second. It was slowed down but this is done with no optimization in place.

In summary, more work needs to be done with the actual hardware to ensure proper performance to meet the requirements of this project.

REFERENCES

1. ©SHARP Corporation. “Datasheet: GP2Y0A21YK0F.” Internet: http://www.trossenrobotics.com/productdocs/GP2Y0A21YK0F.pdf December 1, 2006 [November 18 2011].

2. Eric. “Infrared vs. Ultrasonic - What You Should Know.” Internet: http://www.societyofrobots.com/member_tutorials/node/71 January 2008. [November 18 2011].

3. Tamiya. Internet: http://www.tamiya.com/english/products/58408sx4_wrc/index.htm [November 18 2011].

4. Tower Hobbies. Internet: http://www3.towerhobbies.com/cgi-bin/wti0001p?&I=LXYHY2&P=7 [November 18 2011].

5. Wikipedia. Internet: http://en.wikipedia.org/wiki/BeagleBoard#BeagleBoard-xM [November 19 2011].

6. Arduino. Internet: http://arduino.cc/en/Main/ArduinoBoardMega2560 [November 19 2011].

7. Austriamicrosystems. “Programmable High Speed Magnetic Rotary Encoder” Internet: www.austriamicrosystems.com/AS5030 [November 19 2011].

8."Lane Marking Identification." *Mathworks*. Mathworks, n.d. Web. 5 Jan 2012. <http://www.mathworks.com/matlabcentral/fx_files/10207/1/content/Introduction to MATLAB 7/Demos/LaneMarkings/html/lanemarkings.html> [January 20 2012].

9. Eddins, Steve. "Cell Segmentation." *Steve on Image Processing*. Mathworks, 02 06 2006. Web. 6 Jan. 2012. <http://blogs.mathworks.com/steve/2006/06/02/cell-segmentation/> [January 20 2012].

10. *Image Processing Toolbox*. MathWorks, n.d. Web. 5 Jan 2012. <http://www.mathworks.com/help/toolbox/images/> [January 20 2012].

11. *Neodymium Disc Magnet Field Strengths and Holding Force*. Indigo Instruments. <http://www.indigo.com/magnets/gphmgnts/Nd-disc-magnet-strength-specifications.html> [January 20 2012].