AR:IN

Autonomous Robotics: Intersection Navigation

GROUP 11
System Design

Derek Arts - 1329017
Jonathan Klawunn -1320342
Kate Keskikyla - 1300693
Kerianne Rikley - 1309336
Nabil Hamza - 1317929
Trent Stevenson - 1314633

Table of Contents

List of Figures

List of Tables

Revision History

Table 1: Revision History

Revision Number	Date	Revision Description
0.0	14/12/2016	Document Creation
0.1	21/12/2016	General Work on Document
0.2	23/12/2016	Document Work for System Design 1

Introduction

Purpose

The purpose of this document is to plan and design a system which meets all of the requirements defined in the system requirements document. The document will capture any development decisions for the system's hardware components, as well as document the software structure of the system. This document should clearly define:

1. Any build decisions the development team makes, as well as any issues that may arise
2. The breakdown of system components and their function in the system
3. Standard system operation and system operation in the event of an error
4. System interaction with the surrounding environment

Project Background

AR:IN is the Intersection Navigation team of Autonomous Robotics. Our team focuses on expanding the capabilities of autonomous vehicles to safely navigate intersections. Here at AR:IN passenger safety is the priority and we work to ensure that our product does whatever it can to protect our passengers from harm. Self navigating vehicles will improve traffic flow and have many benefits to the environment.

This project was created for General Motors, who defined this project as:
"Building on past projects that have created solutions for RC cars to be autonomous and follow a laid out path and recognize and avoid obstacles, create a solution where two (or more) such cars can identify an intersection and stop at it, check that all is clear and then proceed. The challenge that additionally presents itself when there are multiple cars is deciding who proceeds first –interpreting the rules of the road and dealing with exceptions in a safe manner." (1)

This document is intended to explain the specifics of our project and project development process for an audience with a technological background, and familiarity with technology. Specifically, this document was created for Dr. Alan Wassyng and all of the Teaching Advisors for the 4TB6 class to give them a comprehensive understanding of our project, as well as our design process. As well, this document was created for the development team to follow when creating the project, to allow for technical specifications to be defined and followed accurately in the build stage.

With the designs outlined in this document, as well as the course timeline and documentation, AR:IN will design and implement the system outlined below, and update this document with any changes to the decisions made.

Scope

Project Goals

- Design a system of 2 (or more) autonomous cars that can:
    - Follow a track
    - Avoid obstacles
    - Recognize and stop at a stop sign
    - Determine who has the right of way
    - Communicate with each other
    - Proceed safely through the intersection when possessing the right of way
    - Follows the rules of the road
    - Maintain reasonable speed while operating
    - Be able to detect critical errors in the system and take measures to prevent further damage to the system
- Stay under budget on project
- Keep system and code simple and effective

Project Deliverables

The key deliverables for this project include a functioning prototype system with two cars that can successfully follow a track, avoid obstacles, recognize and stop at an intersection, as well as appropriately decide who proceeds first from the intersection when multiple vehicles are involved.

Project Costs

Table 2: Project Costs

Item	Estimated Cost
1/10 scale RC road car	$250
Board	$100
Senors	$30
Camera	$50
Batteries	$25
Car to car communication device	TBD
Misc Fasteners/Building Materials	$10
Total	$465 + TBD

Roadmap

List of Expected Changes

1. The states of the system will likely be refined so that redundancies are removed and the system is more efficient
2. Another processor may be added so that the main board will handle image processing and a secondary board to handle motor control.
3. Modules will be added as software becomes refined
4. Variables will be added to increase the depth of the software as testing is done

Unlikely Changes

1. Lane following being controlled by white line sensors instead of image recognition
2. Stopping being taken out of the drive state

Module Guide

Table 3: Module Descriptions and Interactions

Module	Description	I/O	Connected Modules
Main	Controls main operational loop of system, connects to all other modules and calls them when necessary, contains state machine	I → various inputs from all other modules O → state information, calls all other modules	All
Drive	Controls the motor and steering of the vehicle using data detected by other modules	I → information concerning the vehicle's surroundings O → Hardware instructions to the motor and steering	ImageRec → Gets course correction info Sensor → Gets course correction info
Comms	Handles inter-car communication	I → message content from main O → interpretation of received messages from other vehicles	TBD

Sensor	Processes data from an array of external sensors (ex: infrared, sonar, etc.) in order to more accurately analyze the vehicle's surroundings	I → Sensor data O → Boolean on whether or not stop signs, cars or any other obstacles are being detected	Drive → Although not as high a priority as ImageRec, the Sensor module does play a part in dictating how the car moves on the track
ImageRec	Takes an image captured by a camera and processes it to get information of the surroundings. Specifically used for lane tracking, detecting stop signs and traffic awareness	I → Camera image 0 → Boolean for detecting vehicles and stop signs, as well as directions for the steering and motor	Drive → ImageRec sends all outputs to Drive which determines how the vehicle will function

Figure1: Module Connection Diagram

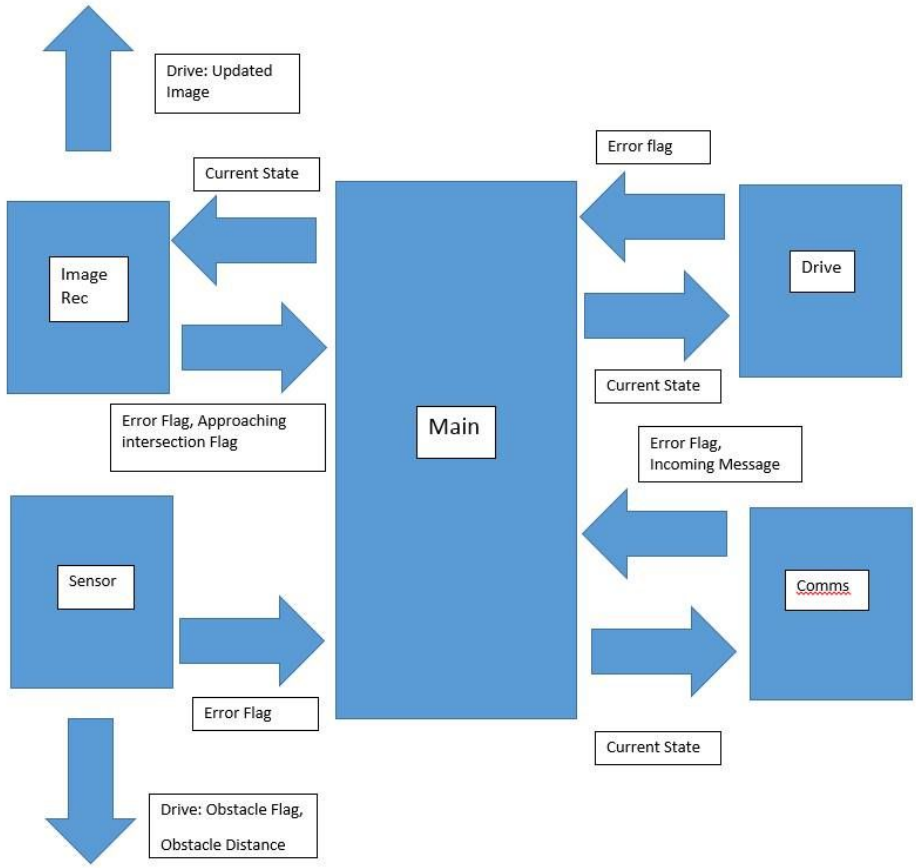


Figure 2: ImageRec module decomposition

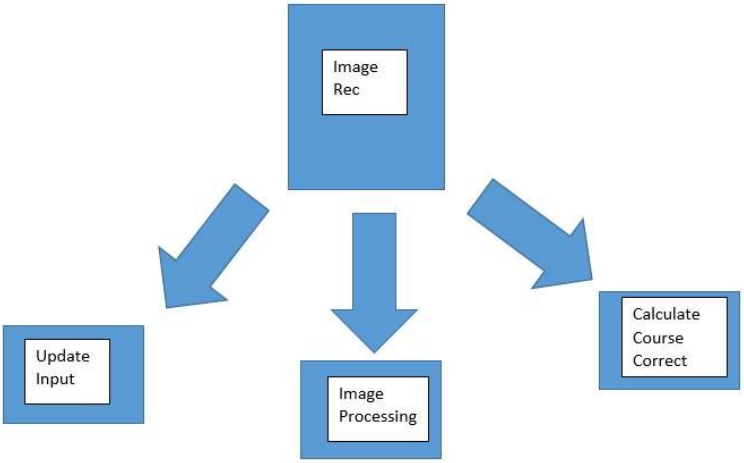


Figure 3: Comms module decomposition

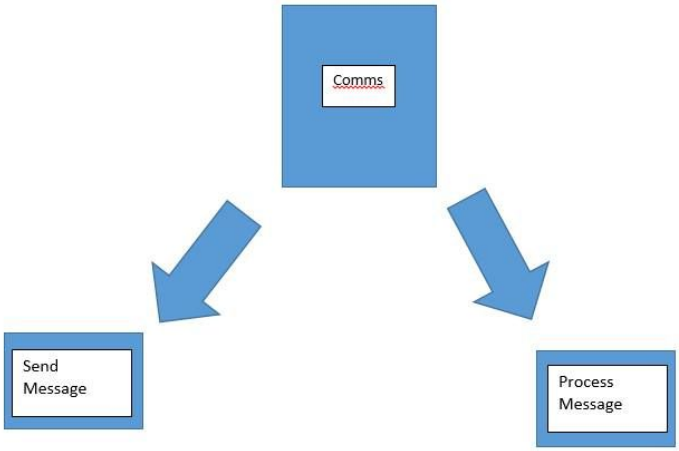


Figure 4: Sensor module decomposition

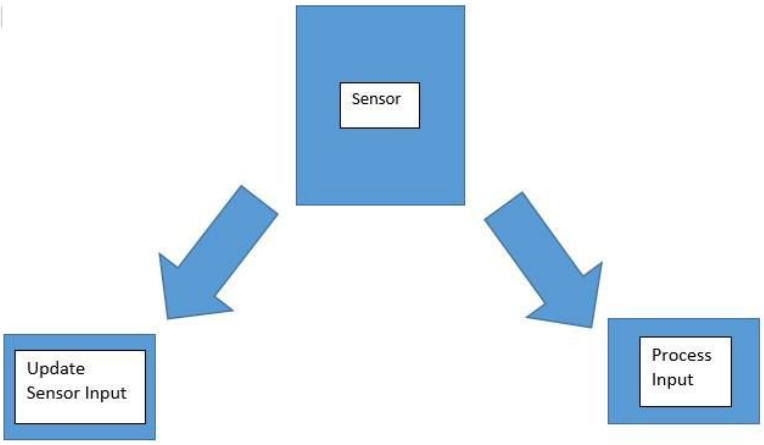
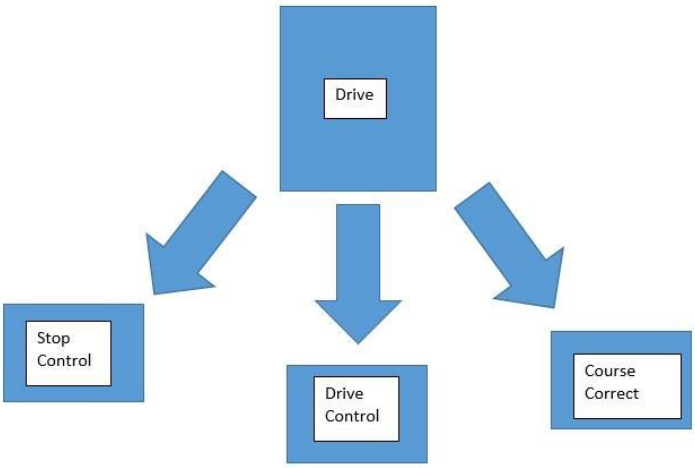


Figure 5: Drive module decomposition



System Decomposition

Software Design

Figure 6: System Decomposition Tree

<table>
<tr><td rowspan="5">Main</td><td rowspan="3">→ Drive</td><td>← Current Voltage</td><td colspan="3" rowspan="2">Motors</td></tr>
<tr><td>→ New Voltage</td></tr>
<tr><td>← Position</td><td>Sensors</td><td>← Voltage</td><td>Sensors</td></tr>
<tr><td>← Position</td><td>ImageRec</td><td>← Image</td><td>Camera</td></tr>
<tr><td>→ Comms</td><td>→ Voltages</td><td>S/R HW</td><td>←→ Message</td><td>Other Cars</td></tr>
</table>

Figure 7: System State Diagram

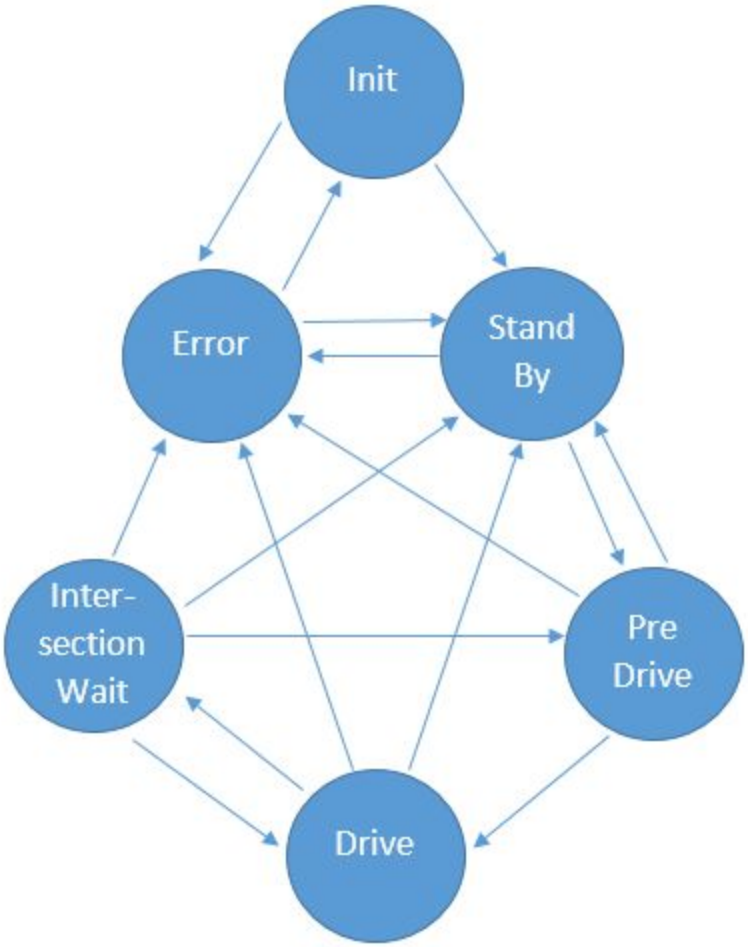


Table 4: State Descriptions

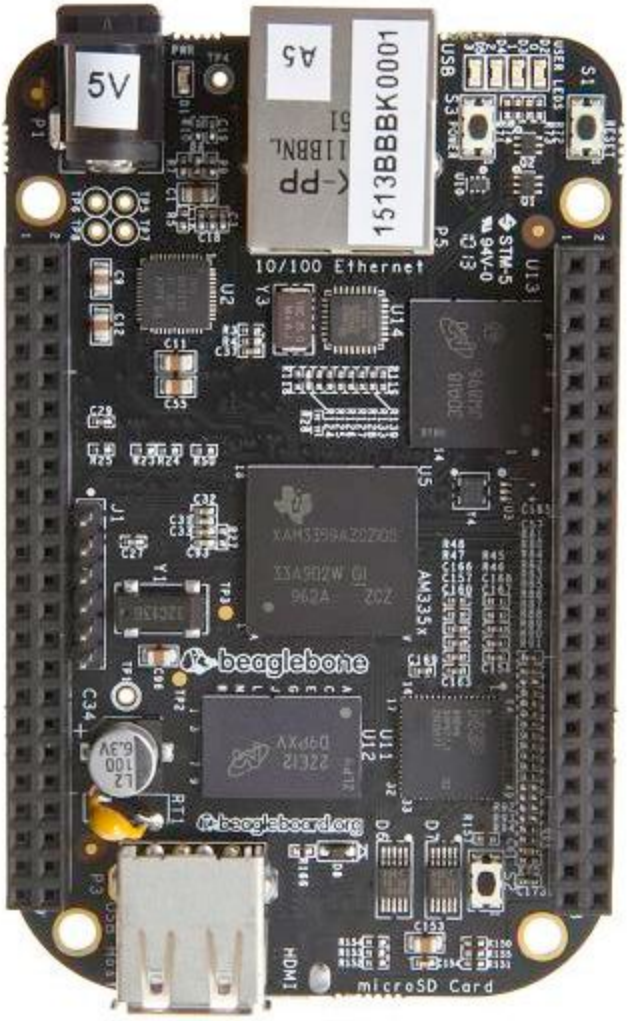
State	Description
Init	In this state the vehicle is stationary and the system is initialized with starting values
Stand-By	In this state the vehicle is stationary and scanning for messages from other vehicles
Pre-Drive	In this state the vehicle its scanning its environment to determine if it is able to clear to enter the drive state
Drive	In this state the vehicle is moving forwards and following lanes while also scanning for intersections and other obstacles. Will adjust speed and stop as needed
Intersection Wait	In this state the vehicle is stationary at an intersection and is sending and receiving messages from other vehicles to decide how to proceed
Error	In this state the vehicle has identified an error and will act based on error. If the error can be repaired the system will attempt to fix it otherwise it will revert to an adaptive drive state

Hardware Design

Figure 8: Vehicle Chassis[2]

This chassis will be able support the weight of the vehicle along with the weight of the software that will be added. Up to two micro processors, a camera, and other smaller sensors will have to be mounted onto the vehicle so the car will have to have enough strength and power to operate with the added weight. The vehicle has to be able to speed up and slow down at the desired rate while also being able to turn safely. The suspension system will help stabilize the vehicle so that the camera will capture clear images and connections will be shook loose. There is enough area on the base of the vehicle to mount the other pieces in an organized fashion.

Figure 9: Beaglebone Black[3]



The microprocessor that will be used is the BeagleBone Black. It will take care of the image recognition, communication, and motor controls. If the board cannot compute these in a timely manner a second board may have to handle the motor control because the computations need to be completed quickly enough to be useful. This board has all the ports that will be needed for the motor control and image processing and likely will for the communication device once it is decided upon.

MIS and MID

Module: Main

MIS

Contains state machine and main execution loop

Interface

Uses

Drive
Sensor
Comms
ImageRec

Type

TBD

Access Programs

getSensorInput: TBD
getErrorFlags: TBD
getStopSignFlag: TBD

MID

Implementation

Var

TBD

Module: Drive

MIS

Top level module that contains drive control, stop control and course correct module used for controlling vehicle movement

Interface

Uses

ImageRec
Sensors
DriveControl
StopControl
CourseCorrect

Type

TBD

Access Programs

getSpeed: TBD
setAcc: TBD
getState: TBD
getCenter: TBD
getInDist: TBD
setDir: TBD
getErrorFlags: TBD
setErrorFlags: TBD

MID

Implementation

Var

TBD

Module: ImageRec

MIS

Top level module containing image capture, image processing and course correction calculation modules used for lane following and obstacle detection

Interface

Uses

UpdateInputImage
ProcessImage
CalcCourseCorrect

Type

TBD

Access Programsyes

setErrorFlags: TBD
getErrorFlags: TBD
setStopSignFlag: TBD

MID

Implementation

Var

TBD

Module: Comms

MIS

Top level module that contains SendMessage and ProcessMessage modules which are used inter-car communication

Interface

Uses

SendMessage
ProcessMessage

Type

TBD

Access Programs

getState: TBD
getMsg: TBD
setMsg: TBD

MID

Implementation

Var

TBD

Module: Sensors

MIS

Top level module containing UpdateSensorInput and ProcessSensorInput modules used for getting sensor input from secondary sensors and transmitting data to Drive module for course corrections.

Interface

Uses

UpdateSensorInput
ProcessSensorInput

Type

Access Programs

getSensorInput: TBD
getState: TBD
setCenter: TBD

MID

Implementation

Var

Bibliography

[1] General Motors of Canada. “GM Autonomous Project.”Internet: https://avenue.cllmcmaster.ca/d2l/le/content/187464/viewContent/1553354/View, [21/12/2016]
[2] Spektrum. “2012 Chevrolet Camaro ZL1 V100-S 1/10 RTR”. Internet: http://www.spektrumrc.com/Products/Default.aspx?ProdID=VTR03007
[3] BeagleBone Black. Internet: https://beagleboard.org/black

Appendix A

Table 5: Constants and Variables

Description of Value	Name***	Type	Units	External
Obstacle Locations	m_OBSLOC	Monitored	m with rad	TBD
Distance to Intersection	m_INDIST	Monitored	m	TBD
Speed	m_SPEED	Monitored	m/s	TBD
Acceleration	c_ACCLRN	Controlled	m/s2	TBD
Safe Stopping Distance	k_SSDIST	Constant	m	TBD
Stopping Flag	y_STPSGN**	Controlled and Monitored	none	TBD
Center of Lane	m_CNTRLN	Monitored	rad	TBD
Direction	c_DIRECT	Controlled	Rad (deg)	TBD
Other Car Outputs	m_OUTxxx*	Monitored	none	TBD
General Outputs	c_OUTxxx*	Controlled	none	TBD
Outgoing Messages	c_MSGOUT	Controlled	none	TBD
Incoming Messages	m_MSGINC	Monitored	none	TBD
Car ID	k_ID	Contant	none	TBD
State	c_STATE	Controlled	none	TBD
Error Flags	y_ERRORx**	Controlled and Monitored	none	TBD

*The x's here are to be replaced with a three character code identifying which output
**The y can be replaced by either an m or a c depending on the module and use. Both an m and a c error flag can exist with a module. The x is to be replace with a number allowing for multiple error flags to exist within a given module.
***An ext_ will be added to the beginning of all external variables and constants.