VIC

Vehicle Intersection Control

McMaster University

System Design

SE 4G06

Group 6

Alex Jackson	-	1302526
Jean Lucas Ferreira	-	1152120
Justin Kapinski	-	1305257
Mathew Hobers	-	1228607
Radhika Sharma	-	1150430
Zachary Bazen	-	1200979

Table of Contents

List of Tables

List of Figures

1 Revisions

Table 1: VIC Table of Revisions

Date	Revision Number	Authors	Comments
March 25, 2017	Revision 1	Alex Jackson Jean Lucas Ferreira Justin Kapinski Mathew Hobers Radhika Sharma Zachary Bazen	- Updated system component information - Added unexpected even handling - Updated system component information - Updated MIS and MID information - Added circuit diagrams
March 24, 2017	Revision 1	Alex Jackson Jean Lucas Ferreira Justin Kapinski Mathew Hobers Radhika Sharma Zachary Bazen	- Updated module numbering - Added module to requirement traceability matrix - Added module change likelihood and ways to change
March 20, 2017	Revision 1	Alex Jackson Jean Lucas Ferreira Justin Kapinski Mathew Hobers Radhika Sharma Zachary Bazen	- Combined phase 1 documents and phase 2 documents - Split MIS and MID into two distinct sections
January 25, 2017	Revision 0	Alex Jackson Jean Lucas Ferreira Justin Kapinski Mathew Hobers Radhika Sharma Zachary Bazen	- System design phase 2
December 21, 2016	Revision 0	Alex Jackson Jean Lucas Ferreira Justin Kapinski Mathew Hobers Radhika Sharma Zachary Bazen	- System design phase 1

2 Introduction

2.1 Document Purpose

Vehicle Intersection Controller (VIC) is a system that allows autonomous cars to proceed through four way stop intersections when the vehicles arrive. The purpose of this document is to provide a comprehensive system overview of the VIC system. In addition, it is intended to provide comprehensive subsystem details that will allow the system to be implemented.

2.2 System Scope

VIC will focus on solving the aforementioned problem on a controlled, indoor track. Autonomous vehicles, using a $\frac{1}{10}$ scale, will be used to simulate real world autonomous cars. To prevent damage of hardware, the autonomous vehicles will be able to detect obstacles. VIC will ignore situations involving non-autonomous cars.

2.3 Document Overview and Intended Audience

This document contains multiple sections that will provide information on: system assumptions and constraints, overall system information, descriptions and diagrams of system components, detailed module interface and design information along with sequence diagrams.

The system overview contains a natural language description of the VIC system and appropriate context diagrams. In the system components section a high level description is given of the inputs and outputs of the VIC system. This is followed by a module guide.

The module guide provides an overview of all the system modules and a natural language description of each module. Each module description includes information on intended behaviour, inputs, outputs, initialization and timing constraints. In addition, a traceability matrix to requirements is included for all modules and the likelihood of change for each module.

This is followed by the Module Interface Specification and Module Internal Design sections. These sections provide detailed information on VIC module interfaces and internal designs. The final section contains system sequence diagrams.

When reading this document it will be helpful to note that there are two main components to VIC. These two components are the intersection component and the vehicle component. Sections are organized with this division.

The intended audience for this document is Sean Marshall (the engineering team leader at GM) who proposed the problem, Dr. Alan Wassyng and the teaching assistants as supervisors of the project, and ourselves as designers of the system.

2.4 Acronyms

Table 2: Acronyms

VIC	Vehicle Intersection Control
IC	Intersection Controller
VC	Vehicle Controller

2.5 Definitions

Table 3: Definitions

VIC	The entire system including the intersection controller, the vehicles, and their corresponding controllers.
IC	The Intersection Controller is the system that tracks the arrival and departure of the vehicles, as well as determining the order in which the vehicles must proceed through the intersection.
VC	The Vehicle Controller is the system that will allow the 1/10 scale RC car to follow lanes, maintain a desired speed, steer itself, and send requests to the intersection controller.

2.5.1 Naming Conventions

Table 4: Naming Conventions

A#	Assumptions
MC#	Mandated Constraint
ICM#	Intersection Controller Module
VCM#	Vehicle Controller Module
VHM#	Vehicle Hardware Module
IDC#	Intersection design component
VCD#	Vehicle design component

2.5.2 Assumptions

A1	Ideal driving conditions on the track
Rationale	Track is situated indoors

A2	Intersection is a four way stop
Rationale	Different intersection arrangements are beyond the scope of this project

A3	Only autonomous car will be present on the track
Rationale	Only autonomous vehicles are within the scope of the project

A4	Vehicles and intersection controller are pre-Bluetooth paired
Rationale	Speeds up communication by allowing messages to be sent directly to a Bluetooth address without having to first discover them.

A5	Intersection directions will be labeled by a colour
Rationale	Allows direction information to be acquired without additional hardware

2.6 Mandated Constraints

MC1	Vehicles must make a complete stop at the intersection before proceeding through
Rationale	Vehicles must follow the rules of the road

MC2	The cars must not turn at the intersection
Rationale	To simplify intersection navigation

3 System Overview

3.1 Behavior Overview

VIC is a system that controls autonomous vehicle traffic flow at an intersection. It consists of two main components, the intersection controller and the autonomous vehicles. These components communicate over Bluetooth communication.

The intersection controller uses a camera to monitor the state of the intersection. Once a vehicle is detected the next intersection state is determined using current autonomous vehicle input and camera data. When the next intersection state is determined, the intersection controller communicates to the appropriate vehicle to proceed. After the vehicle has left, the current intersection state is updated and the next state calculated.

The autonomous vehicles uses a camera to detect the current position within the lanes of the track, and built-in hardware components such as a servo, and a speed controller to set and maintain desired speed and turning angles. Furthermore, with the help of the camera it will be able to detect when the vehicle is approaching an intersection, or if an obstacle is present in the current path.
Obstacle detection ensures the safety of the vehicle and the obstacle itself. Obstacles will detected using the on-board camera and image processing algorithms.

The intersection detection also provides a means for interpreting the current direction the car is approaching from. This feature will become an important piece of information for determining whether the car should stop at the intersection, or if it can safely proceed. This decision will be determined by the intersection controller. Once the decision has been finalized, it will signal the vehicle that it may proceed through the intersection.

3.2 Context Diagrams

Figure 1: Intersection Controller Context Diagram

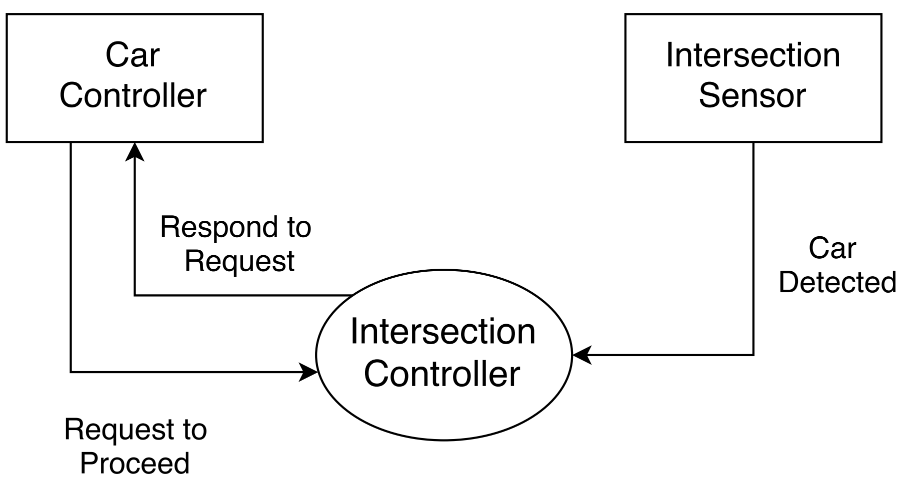
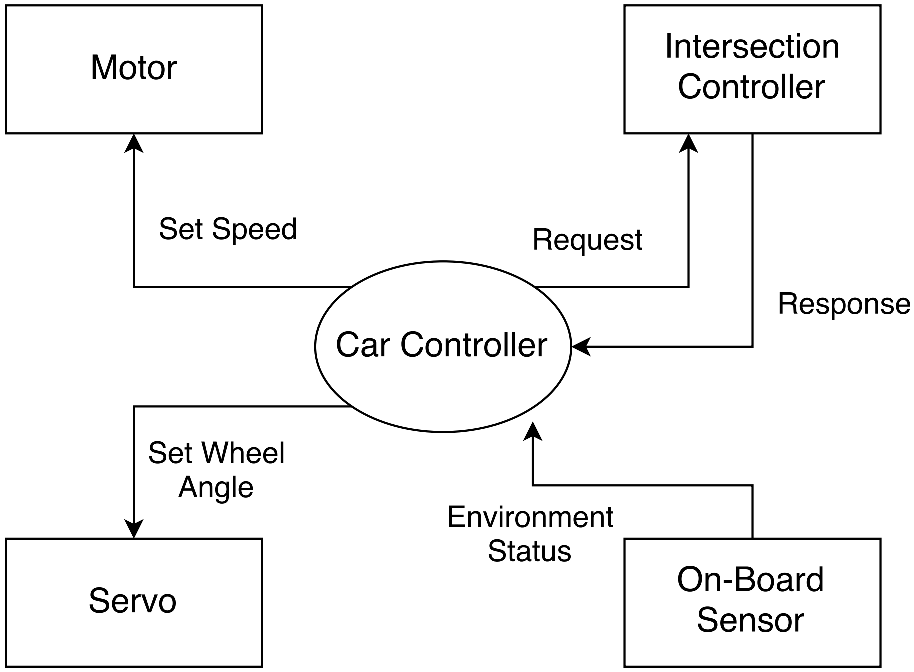


Figure 2: Car Controller Context Diagram



4 System Components

4.1 System Components Description

IDC.1 - Intersection Design Component

Description
This component makes uses of Bluetooth communication and a camera that will be controlled by the Intersection Controller. The intersection controller is responsible for gathering intersection state information and vehicle requests. It uses this information to determine the order vehicles should proceed through the intersection. Once the intersection has been determined to be safe, the system sends a proceed command to the respective vehicle.

There are two external inputs to this component. Firstly, a human input, that will allow the initialization of the intersection controller by turning it on. Secondly, a Bluetooth signal, which will occur during run time when a vehicle sends a request ot notify its intent to drive through the intersection.
Upon initialization, the intersection controller will initialize all required modules and starts the Bluetooth communication.

VDC.1 - Vehicle Design Component

Description
This component makes use of a $\frac{1}{10}$ scale, electric car that will be controlled by the Vehicle Controller. The vehicle controller is responsible for gathering and interpreting vehicle environment information. This information is used to control the vehicles behaviour. The vehicle will follow the lanes of the track, and stop when an intersection or obstacle is present, by utilizing the servo and speed controller of vehicle. Furthermore, when it begins to approach an intersection, it will send a request to proceed to the intersection controller. Once the intersection controller has approved the request, the vehicle will continue its path.

This component will have two sources of external inputs: one human and one Bluetooth signal. The first input is the human interaction that turns on the vehicle, and initializes the vehicle controller. The second input will occur when the vehicle is operational, and receives a Bluetooth signal from the Intersection Controller with some desired value stating an action that vehicle should take.

Upon initialization, the vehicle controller will initialize all required modules, and keep the vehicle in a stopped position until a go-ahead signal is given.

4.2 System Components Diagram

Figure 3: System Component Diagram - Vehicle Component

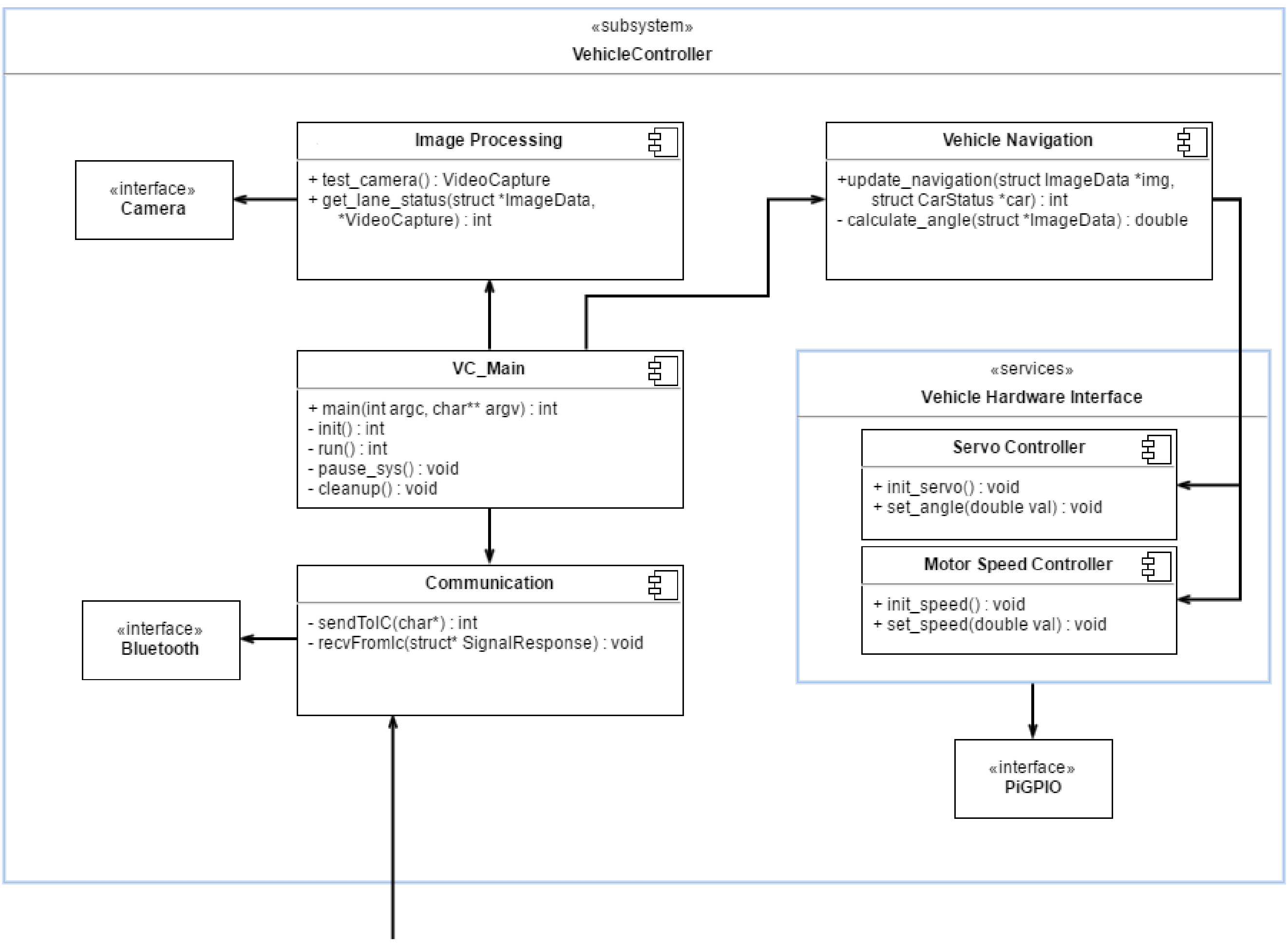
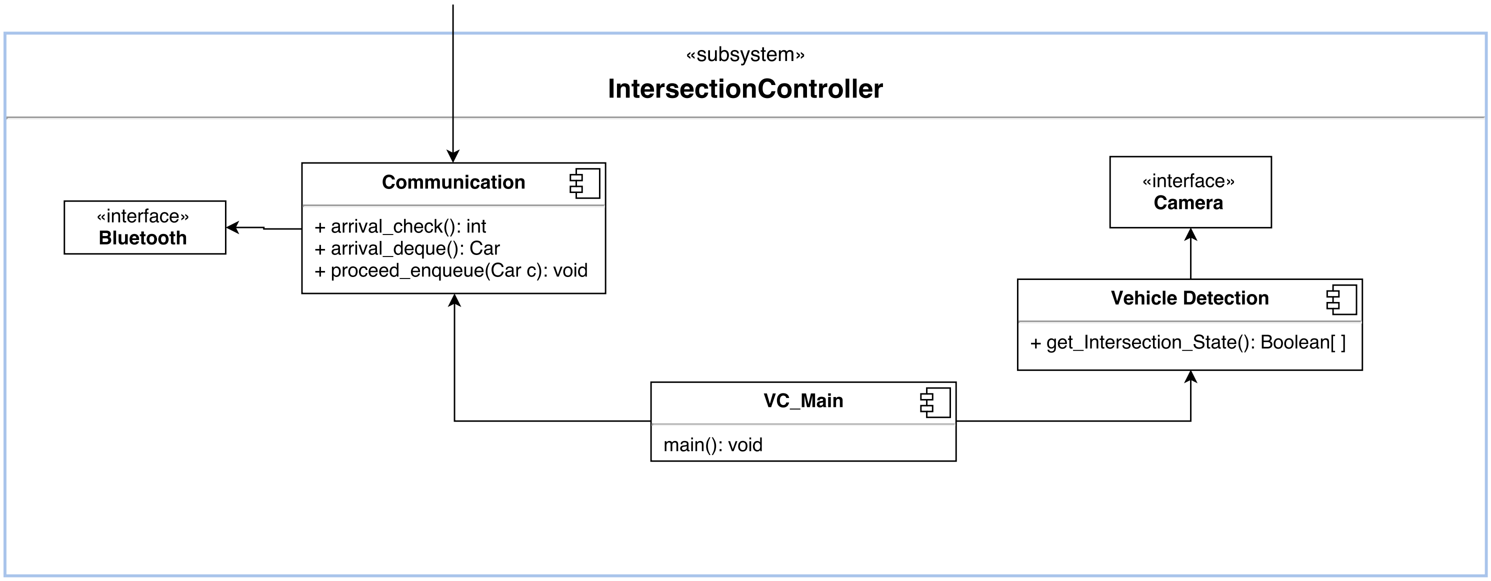


Figure 4: System Component Diagram



5 Module Guide

5.1 Module Overview

5.1.1 Intersection Controller Modules

ID	Name	Responsibilities	Secrets
ICM.1	VehicleDetection	Know when a car is within intersection area	Relationship between intersection camera and car
ICM.2	Communication	Interpret receiving car signals and sending signals to a car	Communication protocol
ICM.3	IC_Main	Determine order of vehicle progression based in input data	Scheduling algorithm

5.1.2 Vehicle Controller Software Modules

ID	Name	Responsibilities	Secrets
VCM.1	ImageProcessing	Interpret image into environment state	Image processing algorithm
VCM.2	VehicleNavigaton	Control the navigation of the car	How the car navigates on the track
VCM.3	Communication	Interpret signal from Intersection Controller. Prepare and send signal to the Intersection Controller	Communication Protocol
VCM.4	VC_Main	Control information flow of the car	Manage car modules

5.1.3 Vehicle Controller Hardware Modules

ID	Name	Responsibilities	Secrets
VHM.1	HAL	The Hardware Abstraction Layer (HAL) setups all necessary hardware modules and libraries	Hardware details
VHM.2	ServoController	Set a physical wheel angle	How to convert a software value to a PWM (Pulse Width Modulation) signal
VHM.3	MotorSpeedController	Control PWM signal	How to convert speed into a PWM signal

5.2 Intersection Controller Software Module Description

ICM.1 - VehicleDetection

Behavioural Description
Vehicle Detection will make use of a camera to view the intersection, from a bird's eye view. It creates and updates a data structure which will hold the state of the intersection.

Inputs
Video stream inputs.

Outputs
VehicleDetection will output a list of the cars that have left the intersection, and whether or not the intersection is occupied.

Initialization Description
IC_Main will initialize this module. Upon initialization, VehicleDetection will turn on the web camera.

Derived Timing Constraints
Time constraint for VehicleDetection must be bounded by the time constraint set by the IC_Main.

Unexpected Event Handling
In the case that the camera becomes disconnected from the system the VehicleDetection will set the intersection state to unknown. This will signal to the IC_Main that there is an error. The system will the stop execution. Without the intersection state the system can not safely schedule vehicle requests.

ICM.2 - Communication

Behavioural Description It will be responsible for the two-way communication from the car to the intersection controller. It will create a list of communications coming in and coming out.

Inputs
A Bluetooth signal containing information from the requesting vehicle. This signal contains information where the vehicle is coming from, going to and a message return port.

Outputs
The module will have two outputs. One will the information of inbound cars to the IC_Main, the other will be a response to the cars to proceed.

Initialization Description

The inbound and outbound list of cars will be initialized empty, and the module will remain running continuously.
A listening socket will be instantiated and the communication protocol set to RFCOMM (Radio Frequency Communication).

Derived Timing Constraints
Limited to the speed of the Bluetooth connection.

Unexpected Event Handling
In the event of an incoming message is incorrect or corrupted the message will be dropped.
In the event where a proceed message is unable to be sent to the car the system will reattempt to retransmit the message. If the message is unable to be sent after a set number of attempts the message will be dropped. This will be communicated to the IC_Main where appropriate action will be taken.

ICM.3 - IC_Main

Behavioural Description
It will be responsible for determining the order vehicles proceed through the intersection. It maintains a list of current vehicles in the intersection, and updates accordingly.

Inputs
Inbound communication from the Communications module, the current state of the intersection from the VehicleDetection module.

Outputs
The output will be cars going into the outbound list of cars that may proceed through the intersection.

Initialization Description
Ensure the list of active cars in the intersection is cleared. It also initializes all other modules connected to it.

Derived Timing Constraints
The timing constraint is based on the speed of the car, and the length of the intersection. In order to detect when a car has entered and left the intersection, we must process the current intersection state within this given time.

$$processing\ Time = \frac{intersectionSegmentLength * intersectionLength}{car\ Speed}$$
$$processing\ Time = \frac{\frac{1}{6} * 0.6\ m}{1.4\ m/s}$$
$$processing\ Time = 0.07\ seconds$$

Unexpected Event Handling
In the event that the intersection state becomes unknown the system will be halted. Without appropriate intersection state information the vehicles can not be safely scheduled.
In the event that the IC_Main is notified that a proceed command could not be communicated to a vehicle the system will then assume that the vehicle is not working and will continue to schedule vehicles on the other unblocked direction.

5.3 Vehicle Software Module Description

VCM.1 - ImageProcessing

Behavioural Description
The image processing module of the vehicle controller is responsible for detecting lane positioning and obstacles. Once these factors have been detected it must produce digital information pertaining to the state of the track as seen in the current image.

Inputs
Video camera stream instance.
Structure containing data from previous output of this module.

Outputs
Digitally interpreted information regarding the state of the car with respect to the track.

Initialization Description
The image processing module will be initialized once the car has been turned on. It requires that the camera has initiated the video stream prior to this first call of this module.

Derived Timing Constraints
The time constraint for the ImageProcessing is bounded by the time constraint in VC_Main.

Unexpected Event Handling
If the camera has been disconnected from the system, the image processing module will not be completed, and will return an appropriate error value to VC_Main. The system will halt, as it cannot go on without a constant video stream.
In the case that the captured image from the video stream is corrupted, an appropriate error message will be returned, but shall not cause the system to be halted.

VCM.2 - VehicleNavigation

Behavioural Description
The vehicle navigation module of the vehicle controller is responsible for ensuring that the vehicle follows the correct lane on the track, stops at intersections, and avoids obstacles. This module will apply any necessary adjustments in trajectory to ensure that these responsibilities are met.

Inputs
Information regarding the current state of the car.
Information regarding the current state of the track with respect to the car's position.

Outputs
Adjustments to the vehicle's steering, acceleration, and braking necessary to continue following the track.

Initialization Description
The vehicle navigation module will be initialized once the vehicle has been turned on. The vehicle's steering angle, speed, and acceleration with be initialized at zero (i.e. standing still with the steering pointed directly forwards).

Derived Timing Constraints
The vehicle navigation module timing constraint is bounded by the contraint set by VC_Main.

Unexpected Event Handling
In the case of an unexpected event such as data corruption in the input, the module may set unsafe car adjustments, causing the car to steer off-track, or potentially crash into another car. Thus on any unexpected event, the module shall notify VC_Main with an appropriate error message, and the system shall halt safely.

VCM.3 - Communication

Behavioural Description
Responsible for establishing communication from the car to the intersection controller, when it begins to approach an intersection. It will also receive the response from the intersection controller if it can proceed through the intersection.

Inputs
Formatted message that is to be sent to the Intersection Controller (on request).
Bluetooth signal from the Intersection Controller (on response).

Outputs
One output will be the information about the car to the intersection communication module. The other will be the response value from the intersection controller to the VC_Main.

Initialization Description
Will be initialized when the car is initiated, and paired to the intersection controller.
The communication protocol will be set to RFCOMM (Radio Frequency Communication), providing a reliable *TCP-like* connection.

Derived Timing Constraints
The timing constraints will be limited to the speed of the Bluetooth connection. The system will not be blocked while waiting for a response signal, or waiting for a successful sent message.

Unexpected Event Handling
In the case of an unsuccessful message sent, or received, the module will continuously re-attempt until success. If it does not succeed after a set number of tries, an error message will be returned to VC_Main, and the system will be halted. Without a reliable communication stream between the Intersection Controller and the Vehicle Controller, we cannot continue to operate, as it may lead to the collision of the cars at the intersection.

VCM.4 - VC_Main

Behavioural Description
This module is responsible for maintaining an active loop while the car is in operation. It will gather information from the ImageProcessing and Communication modules, and transpose that information to the VehicleNavigation module to make certain decision.

Inputs
Formatted Bluetooth signals from the car Communication module.
Digital information pertaining to the state of tract with respect to the car's position.

Outputs
A request for the car Communication module to signal the intersection controller.

Initialization Description
Upon initialization, VC_Main will initialize: the VehicleNavigation module, the Communication Module, and start a video stream instance to be used by the ImageProcessing module.

Derived Timing Constraints
The exact car speed is unknown, but it will be estimated to be 1.4 m/s, and we will require to get an update on the track status every 3cm to maintain accuracy. Therefore, at this speed and this update rate, the VC_Main must process all necessary information within the following time:

$$
\begin{aligned}
processing\ time &= \frac{distance\ update\ rate}{car\ speed} \\
processing\ time &= \frac{0.03\ m}{1.4\ m/s} \\
processing\ time &= 0.02\ seconds
\end{aligned}
$$

This means that the ImageProcessing module and VehicleNavigation module must complete one full iteration within this constrained time.

Unexpected Event Handling
The unexpected events in the other modules have already been described earlier. Once VC_Main receives one of the unexpected event messages, it will decide the appropriate course of action. In some cases, the system shall be deemed unsafe to continue, and thus halted.
In the case that VC_Main itself experiences an unexpected event, it may lead to some module not being called when required. Therefore it is necessary to halt the system in the case of an unexpected event raised in this module.

5.4 Vehicle Hardware Component Description

Design Notes
None.

VHM.1 - HAL

Behavioural Description
Provides an interface for initalizing all hardware componets of the vehicle.

Inputs
None.

Outputs
None.

VHM.2 - ServoController

Behavioural Description
Given a value, the ServoController will evaluate a PWM signal, and set the servo value with this calculated PWM signal.

Inputs
The desired angle for the servo in software.

Outputs
A physical signal on a GPIO pin telling the servo to go to the desired angle.

VHM.3 - MotorSpeedController

Behavioural Description
Given a value, the MotorSpeedController will evaluate a PWM signal, and set the speed controller value with this calculated PWM signal.

Inputs
The desired speed of the vehicle in software.

Outputs
A physical signal on a GPIO pin telling the motor to go at a specific speed.

6 Module Traceability Matrix

Identifier	Mapped Reqs	V1	V2	V3	V4	V5	V6	IC1	IC2	IC3	IC4	IC5	IC6
Mapped Modules		1	4	2	5	6	5	1	1	1	1	1	1
ICM.1	2							X	X				
ICM.2	2										X	X	
ICM.3	2									X			X
VCM.1	3		X	X			X						
VCM.2	4		X		X	X	X						
VCM.3	1	X											
VCM.4	5		X	X	X	X	X						
VHM.1	2				X	X							
VHM.2	3		X			X	X						
VHM.3	2				X	X							

7 Module Change Likelihood

7.1 Intersection Controller

Module	Change Likelihood	Ways to Change
ICM.1	Very Likely	Physical sensors instead of camera
ICM.2	Very Likely	Different transmitting technology in order to scale
ICM.3	Unlikely	Additional inputs to system

7.2 Vehicle Controller Modules

Module	Change Likelihood	Ways to Change
VCM.1	Very Likely	New image processing algorithm for increased accuracy and reliability
VCM.2	Likely	Feedback controller calibration
VCM.3	Likely	Different transmitting medium in order to scale
VCM.4	Unlikely	Additional inputs to system

7.3 Vehicle Hardware Modules

Module	Change Likelihood	Ways to Change

VHM.1	Unlikely	Change hardware interface library
VHM.2	Unlikely	New servo requires calibration
VHM.3	Unlikely	New speed controller requires calibration

8 Module Interface Specification

8.1 Intersection Module Interface Specification

Table 30: ICM.1 - VehicleDetection

ICM.1 - VehicleDetection	
VehicleDetection()	Constructor to initialize the detection of vehicles at the intersection.
get_intersection_state( ) : String [ ]	Returns the state of the intersection when the function is called. Returns list of cars that have left the intersection and whether the intersection is occupied or not

Table 31: ICM.2 - Communication

ICM.2 - Communication	
Communication()	Constructor to initialize and start communication services
arrival_check() : int	Allows the IC_Main to check if vehicle requests have arrived
arrival_deque(): Car	Function to allow the controller recieve a request to be scheduled from the car.
proceed_enqueue(Car c) : void	Function that allows the intersectrion to send a car the response to proceed through the intersection.

Table 32: ICM.3 - IC_Main

ICM.3 - IC_Main	
main() : void	Main Function for VIC.

8.2 Vehicle Module Interface Specifications

Design Notes
Please see section 9.2 for the definitions of the non-primitive types.

Table 33: VCM.1 - ImageProcessing

VCM.1 - ImageProcessing	
get_lane_status() : ImageData	Function to capture images of the track environment from a webcam and process it into information that can be analysed by software. Will return a defined type of ImageData.
test_camera() : VideoCapture	Confirm the connected camera is functional, and return instance of the video camera object

Table 34: VCM.2 - VehicleNavigation

VCM.2 - VehicleNavigation

update_navigation(struct *ImageData, struct *CarStatus)	Function to signal to the vehicle if there is a change in the navigation, and if so, what changes should be made.

Table 35: VCM.3 - Communication

VCM.3 - Communication	
sendToIC(char* message) : int	Function to allow the car to send the requesting message to the intersection controller. Will return number of bytes sent over. In a successful event, the number of bytes sent should equal number of bytes in the message given as input.
recvFromIC(struct *SignalResponse) : void*	Function to allow the vehicle to receive a response from the intersection controller, once a message is received, the function will update a shared variable of the SignalResponse type, which can be access by VC_Main.

Table 36: VCM.4 - VC_Main

VCM.4 - VC_Main	
main() : int	Function to initiate the vehicle controller.
init() : int	Initiate any necessary modules
run() : int	Enter the program main loop.
pause_system() : void	Stop the car at the current position, and remain inactive until a proceed/continue message is received from the Intersection Controller. Upon which, the system will continue its original behaviour.
cleanup() : void	Free any allocated memory, stop the car, and end any further executions of the system.

8.3 Vehicle Hardware Module Interface Specification

Table 37: VHM.1 - HAL

VHM.1 - HAL	
vichw_init() : int	Initializes PiGPIO library, servo controller and speed controller
vichw_deinit() : void	Terminates PiGPIO library, sets speed and servo PWM signal to zero

Table 38: VHM.2 - ServoController

VHM.2 - ServoController	
vichw_init_servo(void) : void	Initialize servo controller and setup necessary hardware details.
set_angle(angle : double) : void	Outputs a physical signal to the servo to go to the specified angle.

Table 39: VHM.3 - MotorSpeedController

VHM.3 - MotorSpeedController	
vichw_init_speed() : void	Initialize speed controller and setup necessary hardware details.
set_speed(speed : double) : void	Outputs a physical signal to the motor to go at a specified speed.

9 Module Internal Design

9.1 Intersection Module Internal Design

Non-Primitive Types for the Intersection Controller

Car
Holds information pertaining to car communication request.

Fields	Type
direction_from	string
direction_to	string
client_Bluetooth_ID	string
port	int
proceed_now	boolean
retransmission_count	int

ICM.1 - VehicleDetection

Dependancies

- OpenCV - *External library for various image processing features*d
- Numpy - *Python package for scientific computation*

Constants

Name	Type	Description
LowerRed	np.array	Lower bound on red HSV values
UpperRed	np.array	Upper bound on red HSV values
LowerBlue	np.array	Lower bound on blue HSV values
UpperBlue	np.array	Upper bound on blue HSV values
LowerGreen	np.array	Lower bound on green HSV values
UpperGreen	np.array	Upper bound on green HSV values
LowerYellow	np.array	Lower bound on yellow HSV values
UpperYellow	np.array	Upper bound on yellow HSV values

Objects, Macros, Structs, and Types

Name	Defined By	Description
intersection_state	array	Stores current state of the intersection
VideoCapture	OpenCV	Provides an API for handling image and video capturing

Access Methods

Name	Parameters	Description	Return Type
get_intersection_state( )	none	Continuously convert images from the webcam and represents them in an array	string [ ]

ICM.2 - Communication

Dependencies

- bluetooth - *External Bluetooth API library*
- collections - *Provides various data structure APIs*
- threading - *Provides multi-threading features*

Constants

Name	Type	Description
RECIEVE_PORT	int	Stores the host listening port

Objects, Macros, Structs, and Types

Name	Defined By	Description
arrivals	collections	Provides arrival buffer for multiple car requests
proceed	collections	Provides a buffer for pending proceed commands

Access Methods

Name	Parameters	Description	Return Type
init	none	Starts send and receive on independent threads	None
receive	none	Receives vehicle requests and puts requests in arrival queue	None
send	none	Sends proceed commands that are in the proceed queue	None
message_extraction	client_message, client_Bluetooth_ID	Extracts the contents of a vehicle request and creates a Car object	Car
arrival_check	none	Returns 1 if vehicle requests are present	int
arrival_deque	none	Returns car at top of the communication buffer	Car
proceed_enqueue	Car	Appends proceed commands to proceed list	None

ICM.3 - IC_Main

Dependencies
None

Constants
None

Objects, Macros, Structs, and Types

Name	Defined By	Description

intersection_cars	array	Data structure containing the cars currently in the intersection

Access Methods

Name	Parameters	Description	Return Type
check_intersection_state	none	Determines which cars have left the intersection and updates the intersection data structure state accordingly	none
run	none	Method that will facilitate the passing of information to other modules. Will determine when cars should be added or removed from the queue.	none

9.2 Vehicle Module Internal Design

Non-Primitive Types for the Vehicle Controller

ImageData
Holds information pertaining to the data gathered from the image processed by the ImageProcessing module. Defined in *vic_types.h*

For further explanations please refer to section 11 (ImageData Reference)

Fields	Type	Decription
avg_left_angle	double	average angle to detected lines of the left lane from the car center
avg_right_angle	double	average angle to detected lines of the right lane from the car center
left_line_length	double	average distance to detected lines of the left lane from the car center
right_line_length	double	average distance to detected lines of the left lane from the car center
intersection_distance	double	distance to the intersection if detected
intersection_detected	bool	true if intersection has been detected
obstacle_detected	bool	true if obstacle has been detected
trajectory_angle	double	a trajectory angle to reach a target point on centered in the lane

CarStatus
Holds information pertaining to the data about the car. Defined in *vic_types.h*

Fields	Type	Decription
car_id	int	Identification number of the car
current_speed	double	current speed set on the car
current_wheel_angle	double	current servo angle of the car
intersection_stop	bool	true if the car is stopped at an intersection

obstacle_stop	bool	true if the car has stopped due to an obstacle on the way

VideoCapture
An OpenCV object that contain access methods to video stream captured the system's camera. Defined in *opencv2/highgui.cpp*

SignalResponse
Holds information retrieved from the signal sent from the intersection controller to the car communication module. Defined in *vic_types.h*

Fields	**Type**	**Decription**
status	int	Integer value representing the action the car should take

VCM.1 - ImageProcessing

Dependencies

- OpenCV - *External library for various image processing features*
- vector - *API for vector class*
- math.h - *API for mathematical functions*
- image_processing.h *Header file the ImageProcessing interface*
- vic_types.h - *Structures defined to be used by the vehicle modules*

Constants

Name	**Type**	**Description**
DEFAULT_CAMERA_ID	int	Integer value to access the camera
CUTOFF_HEIGHT_FACTOR	double	A percentage amount to crop from the top of the captured image.

Objects, Macros, Structs, and Types

Name	**Defined By**	**Description**
ImageData	vic_types.h	Hold information about the captured image of the track
Point	OpenCV	Describes a two dimensional point with fields x and y
Vector	vector	Many vectors will be used to maintain collection of certain elements
VideoCap	OpenCV	Provides an API for handling image and video capturing
Mat	OpenCV	n-dimensional array for representing image data

Access Methods

Name	**Parameters**	**Description**	**Return Type**

lane_status	struct *ImageData struct *CarStatus	Evaluate and convert captured image into useful information about the state of the car on the track	int
test_camera	VideoCapture	Test whether the camera attached to the car is functional, and return instance of the VideoCapture object	VideoCapture

VCM.2 - VehicleNavigation

Dependencies

- vehicle_navigation.h - *VehicleNavigation interface*
- vic_types.h - *Structures defined to be used by the vehicle modules*
- vic_servo_controller.h - *Interface for servo access methods*
- vic_motor_speed_controller.h - *Interface for speed controller access methods*

Constants

Name	Type	Description
MAX_SPEED	double	Maximum allowed speed of the car
MAX_ANGLE	double	Maximum allowed wheel rotation angle

Objects, Macros, Structs, and Types

Name	Defined By	Description
ImageData	vic_types.h	Hold information about the captured image of the track
CarStatus	vic_types.h	Hold information about the car

Access Methods

Name	Parameters	Description	Return Type
calculate_angle	ImageData img	Calculate an appropriate steering angle based on the ImageData information	double

VCM.3 - Communication

Dependencies

- stdlib.h - *Provides general programming functions*
- bluetooth.h - *External Bluetooth API library*
- rfcomm.h - *API for RFCOOM Bluetooth protocol*
- communications.h - *Communication interface*
- vic_types.h - *Structures defined to be used by the vehicle modules*

Constants

Name	Type	Description
RECEIVE_PORT	int	The port Car Communication will be listening to, for a response from the Intersection Controller
INTERSECTION_HOSTNAME	char*	hostname of the intersection controller

Objects, Macros, Structs, and Types

Name	Defined By	Description
SignalResponse	vic_types.h	Information for the signal received fromthe intersection controller

Access Methods

Name	Parameters	Description	Return Type
sendToIC	char* message	Send signal request to the intersection controller, and return the number of bytes sent successfully	int
recvFromIC	struct *SignalResponse	Receive signal from the intersection controller, and update the shared variable passed by argument. This function will be continuously running, and listening to a response on a separate thread.	void*

VCM.4 - VC_Main

Dependencies

- stdlib.h - *Provides general programming functions*
- communications.h - *Communication module interface*
- image_processing.h - *ImageProcessing module interface*
- vehicle_navigation.h - *VehicleNavigation module interface*
- vic_hardware.h - *Interface for all the hardware modules*
- vic_types.h - *Structures defined to be used by the vehicle modules*
- pthreads.h - *API for multi-threaded program*

Constants

Name	Type	Description
CAR_ID	int	car identification value
PROCEED_RESP	int	integer value representing the car is safe to proceed
STOP_RESP	int	integer value representing the car should stop and enter a pause state
EMERGENCY_STOP_RESP	int	integer value representing the car is to stop immediately, and halt all further executions.

Objects, Macros, Structs, and Types

Name	Defined By	Description

ImageData	vic_types.h	Hold information about the captured image of the track
CarStatus	vic_types.h	Hold information about the car
SignalResponse	vic_types.h	Hold information about the response received from the Intersection Controller
VideoCapture	openCV	Video stream object of the car's camera

Access Methods

Name	**Parameters**	**Description**	**Return Type**
main	none	Function to initiate the vehicle controller.	int
init	none	Initiate any necessary modules	int
run	none	Enter the program main loop	int
pause_sys	none	Pause all executions of the Vehicle Controller until we receive a proceed signal from the Intersection Controller	void
cleanup	none	Free all dynamically allocated memory, and halt the system	void

9.3 Vehicle Hardware Module Internal Design

VHM.1 - HAL

Behavioural Description The Hardware Abstraction Layer (HAL) setups all necessary hardware modules and libraries. It is also responsible for calling the initialization methods for the servo and speed controller.

Dependencies

- vic_hardware.h - *Interface for all the hardware modules*
- pigpios.h - *Interface for PiGPIO library*
- servo_controller.h - *Interface for servo controller*
- motor_speed_controller.h - *Interface for speed controller*

Constants

Name	**Type**	**Description**
DEFAULT_PWM	int	Value that represents a centered PWM
MAX_SPEED_PWN	int	Maximum allowed signal for the speed controller
MIN_SPEED_PWM	int	Minimum allowed signal for the speed controller
MAX_SERVO_PWN	int	Maximum allowed signal for the servo controller
MIN_SERVO_PWM	int	Minimum allowed signal for the servo controller

Access Methods

Name	**Description**
set_PWM(pin_number, duty_cycle) : void	Used to output a PWM signal to a given GPIO pin at a given duty cycle.

VHM.2 - ServoController

Generate physical signals that will control the steering servo of the vehicle.

Dependencies

- vic_hardware.h - *Interface for all the hardware modules*
- pigpios.h - *Interface for PiGPIO library*
- servo_controller.h - *Interface for servo controller*

Constants
None

Access Methods

Name	**Description**
vichw_init_servo() : void	Initialize servo controller and setup necessary hardware details.
set_angle(angle : double) : void	Outputs a physical signal to the servo to go to the specified angle.

VHM.3 - MotorSpeedController

Generate physical signals that will control the motor of the vehicle.

Dependencies

- vic_hardware.h - *Interface for all the hardware modules*
- pigpios.h - *Interface for PiGPIO library*
- motor_speed_controller.h - *Interface for speed controller*

Constants
None

Access Methods

Name	**Description**
vichw_init_speed() : void	Initialize speed controller and setup necessary hardware details.
set_speed(speed : double) : void	Outputs a physical signal to the motor to go at a specified speed.

10 Circuit Diagrams

Figure 5: Circuit Diagram - Raspberry Pi General Purpose Input / Output

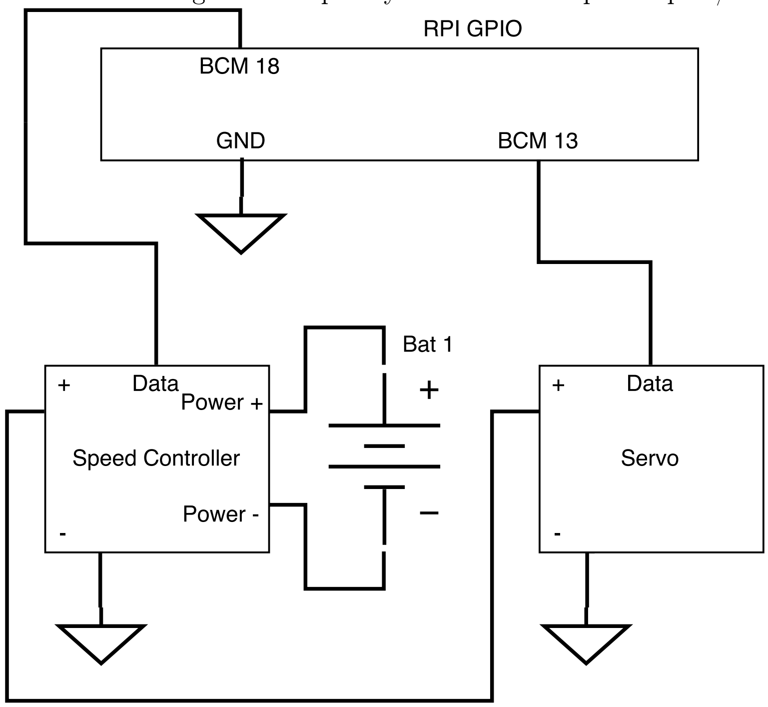
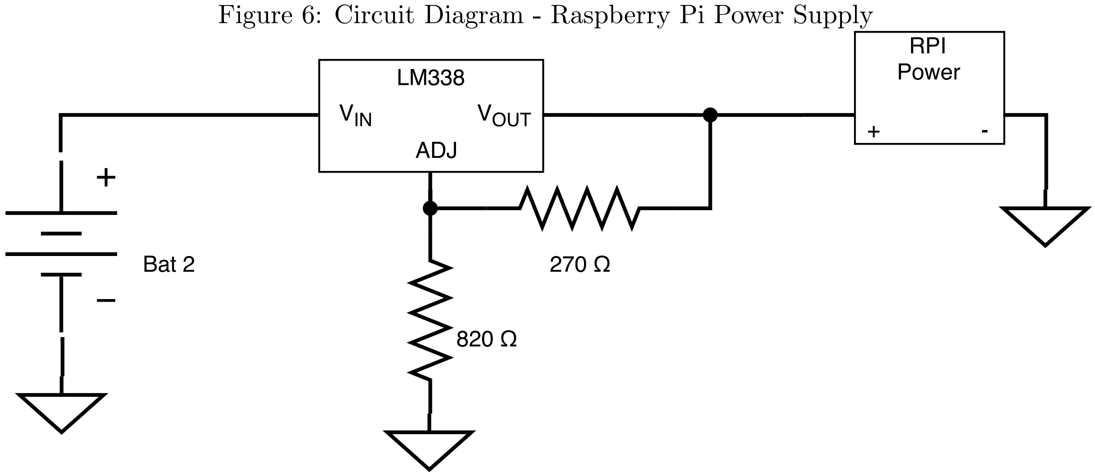


Figure 6: Circuit Diagram - Raspberry Pi Power Supply

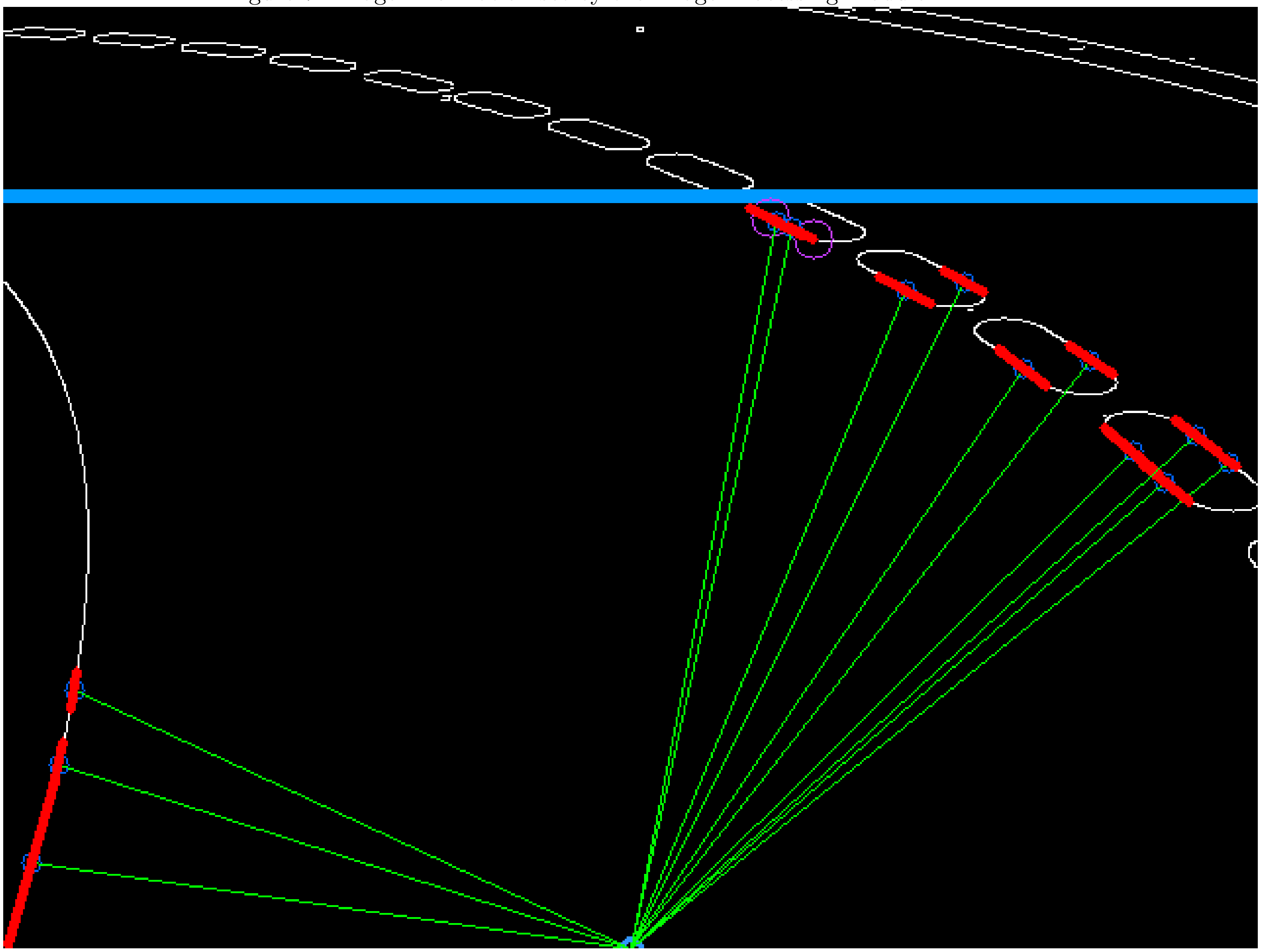


11 ImageData Reference

The following figure provides an example of the image captured and processed by the ImageProcessing module. This will allows us to transform this image into useful data for the ImageData struct.
The blue horizontal line at the top represents the cutoff point for the image, anything above this line is ignored.
The blue circle at the bottom of the image represents the center point of the car.
The yellow circle represents the trajectory point. A desired point the car should attempt to reach.
The green lines to the left of the center point provide a means to calculate the average left angle, and the average left line length. Likewise for the green lines to the right of the center point, allows us to calculate the average right angle, and average right line length.

The white lines were captured through OpenCV's Canny Edge Detector algorithm, and the red lines were processed through OpenCV's Hough Transform for detecting straight lines.

Figure 7: Image Information set by the Image Processing Module



12 Scheduling

Figure 8: Intersection Component Sequence Diagram

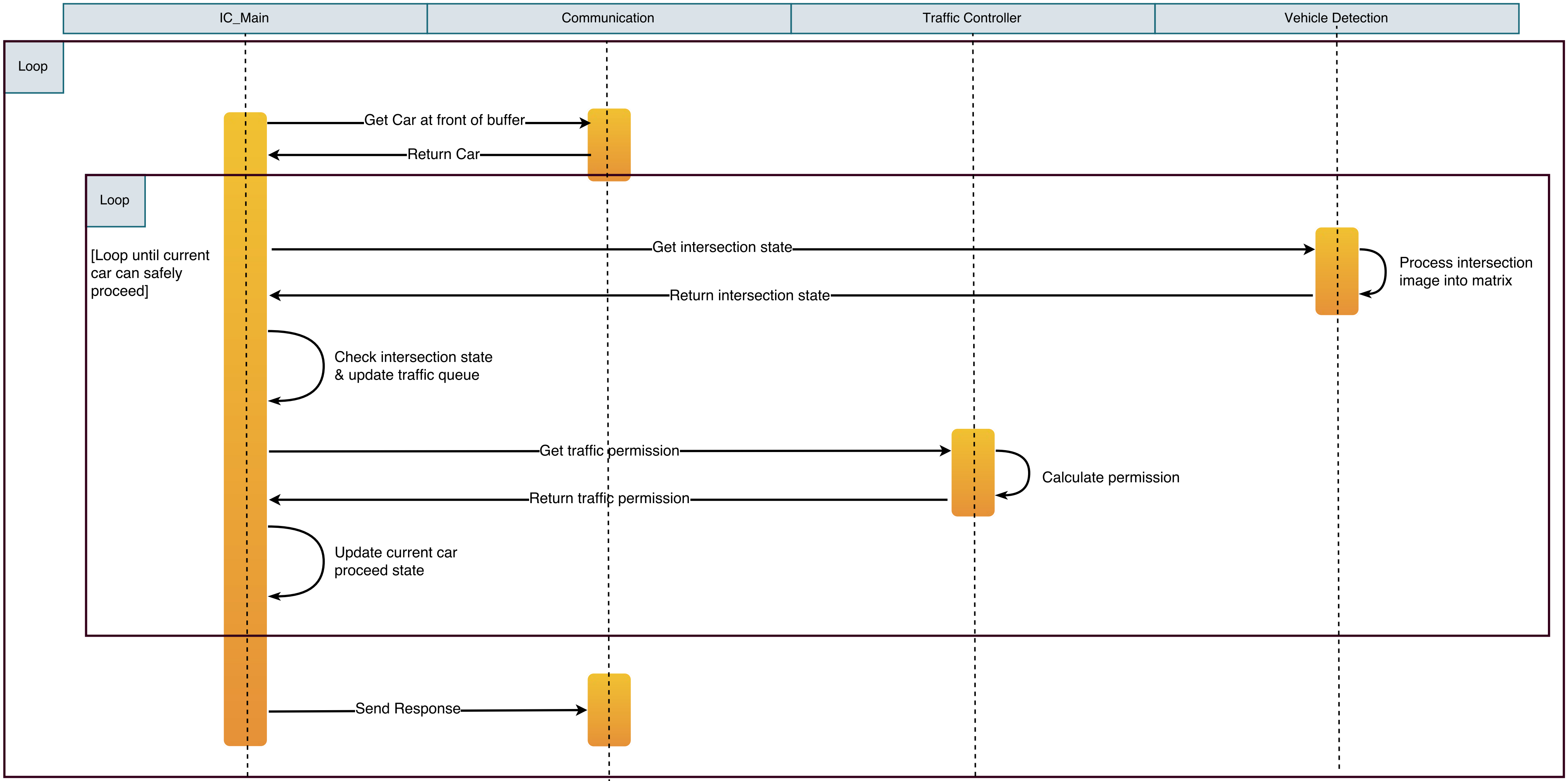


Figure 9: Vehicle Component Sequence Diagram