EEET2162 Design Project

Initial Report

Group 7 -
Dale Giancono, Hanuman Crawford, Alexander Knapik
School of Engineering
RMIT

November 3, 2018

Contents

1 Contribution Declaration 3

2 Introduction 4
2.1 Problem Statement 4

3 Discussion Leading to Initial Design 5
3.1 Assumptions 5
3.2 Primitives Discussion 5
3.3 Fault and Failure Tolerance Discussion 6

4 Design Discussion 8
4.1 Intersection Overview 8
4.2 Context Overview 9
4.3 Use Case Overview 12
4.4 Use Case Scenarios 12
4.4.1 Commuter Use Case 12
4.4.2 Pedestrian Use Case 13
4.4.3 Train Use Case 14
4.4.4 Train Accident Use Case 14
4.4.5 Traffic Accident Use Case 15
4.4.6 Central Control Failure 16
4.4.7 Message Passing Failure 16
4.4.8 Train Controller Node Failure 16
4.4.9 Traffic Controller Node Failure 17
4.5 State Overview 17
4.5.1 Traffic Lights 17
4.5.2 Railway Crossing 19
4.6 Task Architecture Diagram 21
4.7 Messages Passing 23
4.7.1 Traffic Light Messages 23
4.7.2 Railway Crossing Messages 23
4.7.3 Central Control Messages 23
4.8 Node Overview 23
4.9 Node I/O 24

5 Conclusion 28

1 Contribution Declaration

For this RMIT initial report submission for EET2162 Real Time Systems, we as a group including Dale Giancono, Hanuman Crawdord, and Alexander Knapik make the following declaration:

- This work is our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
- No part of this work has been written for us by any other person.
- We have not previously submitted this work for this or any other course.
- All members of the group have contributed to the submission in an equal and fair manner.

2 Introduction

2.1 Problem Statement

A system is required to manage a busy traffic intersection modelled after a real world system of two traffic intersections, consisting of three roads where in between the two traffic intersections, there will be a train crossing including a boom gate with flashing lights to stop traffic from passing simultaneously with a train. Each traffic light state change shall communicate accordingly with a central controller to allow for constant monitoring.

The roads consist of two-way traffic with two lanes in each direction. At each intersection, traffic should be able to cross and turn from each position, and there shall be support for pedestrian crossings. Furthermore, two of roads have a greater chance of congestion out of the three due to being major roads; the specifics of which are detailed in figure 1.

The train crossing's boom gate is to communicate with the adjacent traffic light controller nodes about oncoming trains to alleviate traffic blockage, however the traffic light controller nodes are to be operated separately. If there is a fault with the boom gate not correctly opening or closing, the train crossing's controller node shall communicate with with the central controller about the fault.

A central controller will be available for monitoring the status all nodes within the network, as well as controlling the nodes when human interaction is required for traffic congestion, safety, or legal reasons. However, none of the intersection nodes in the network should depend on the central controller for normal operation, ensuring there is not a single point of failure within the system.

All controller nodes, and the central controller are to be programmed utilising the C programming language and for the QNX real-time operating system.

3 Discussion Leading to Initial Design

3.1 Assumptions

The decisions made which will be discussed in the following section are based on some assumptions on top of the initial problem statement. Firstly regarding the traffic lights, with the two lanes in each direction, there will not be any turn signals at the traffic intersections. Rather there will just standard green lights, and cars can turn freely whenever there is no opposing traffic. The speed limit for the roads will be design for city traffic, and not be greater than 60km/h.

The train railroad will have trains coming in from both directions, and the boom gates will only be activated by sensors on the railroad, not a timetable due to the unpredictable nature of public transport. Furthermore, there will not be any pedestrian crossings through the intersection between the railway and the road.

The local traffic and train nodes will be running on an ARMv7 instruction set, and therefore won't be binary compatible with an x86 architecture.

3.2 Primitives Discussion

In regards to both between intra-process and inter-nodal communication, the following primitives were chosen for utmost safety. Managing a traffic light system, especially with a train line crossing the road requires deep thinking on how to maintain safety even with partial system failure, as a fault within the system not handled properly, can lead to the death of either a commuter, pedestrian or train passenger.

It was decided that within the same process that data should be bundled and shared via using C structures, passed by reference, and only if access to the structure is required by the **entire** process will it be declared globally.

Shared data required to be accessed by more than one thread simultaneously will be required to be locked with a *timed* read-write lock. This allows for error checking and a fail safe, ensuring that even if a thread has crashed or is locked-up, that the process can handle its failure, without locking out the data completely for all other threads. Furthermore, if a thread failed while writing a data-set under a read-write lock, then the threads reading aren't locked out from reading. Mutual exclusions are faster within QNX as they make up the read-write lock, however the extra redundancy is used to ensure a hard real time system is met.

Communication between processes within and outside of the same node will be handled via QNX's native message passing system. These messages can be sent via the QNET network to other nodes, allowing for communication between the central controller and all the other different local intersection nodes. Furthermore, the QNX native message passing requires a reply after a message is sent, therefore error checking can be applied to determine whether the network is correctly operating, or if there is a fault with a specific node's connection.

3.3 Fault and Failure Tolerance Discussion

Road and traffic accidents are one of the leading causes of death within the world, therefore it is imperative that the systems designed for the public road network is as safe as possible.

Faults regarding state changes within the traffic light and train controller nodes are not allowed whatsoever, as a fatal accident could occur if for example the traffic lights were to skip a step. Therefore, all state changes are to be grey encoded as an extra precaution.

The nodes themselves should not be dependant on any other nodes to function correctly and should be completely self sufficient, this way there will not be any single point of failure within the system. This also includes the central controller which should only be available to monitor and set modes of operation, but never to actively control any other nodes to ensure complete operation of all nodes in the case of a central controller failure.

Sending messages from one node to another node however is the action most likely to fail on occasion within the system, so some precautions should be taken in order to ensure that the action fails the least amount of times as possible. To ensure that a recipient receives and replies from the messages passed, if no reply is received, the sender should try again a number of times before deciding that it is unable to get a message from the node that it needs to, and continues.

The traffic lights nodes are constantly sending status updates every time there is a state change. Due to the large amount of status updates that will be sent to the central controller, there will be dropped messages from time to time. Although helpful to those managing the central controller, helping alleviate congestion, these messages sent should never be essential to the operation of the traffic lights.

Messages sent to the traffic light controller from the central controller regarding scheduling changes are not essential the safety, however these messages dropped can be seen as an annoyance to those operating the central computer.

Messages regarding signalling and boom gate faults with the train crossing node are essential to be sent to the central controller. However, there is still the possibility that there may also be a networking fault, and therefore if the central controller is unable to receive any of the messages from the train controller, there must be a fail safe measure for the train conductors, signalling that they are not to cross the intersection until the faults are cleared.

4 Design Discussion

4.1 Intersection Overview

The intersection that the project will be built around involves two sets of traffic lights with a dual train line nested between the traffic lights. The traffic lights are situated on four way intersections.

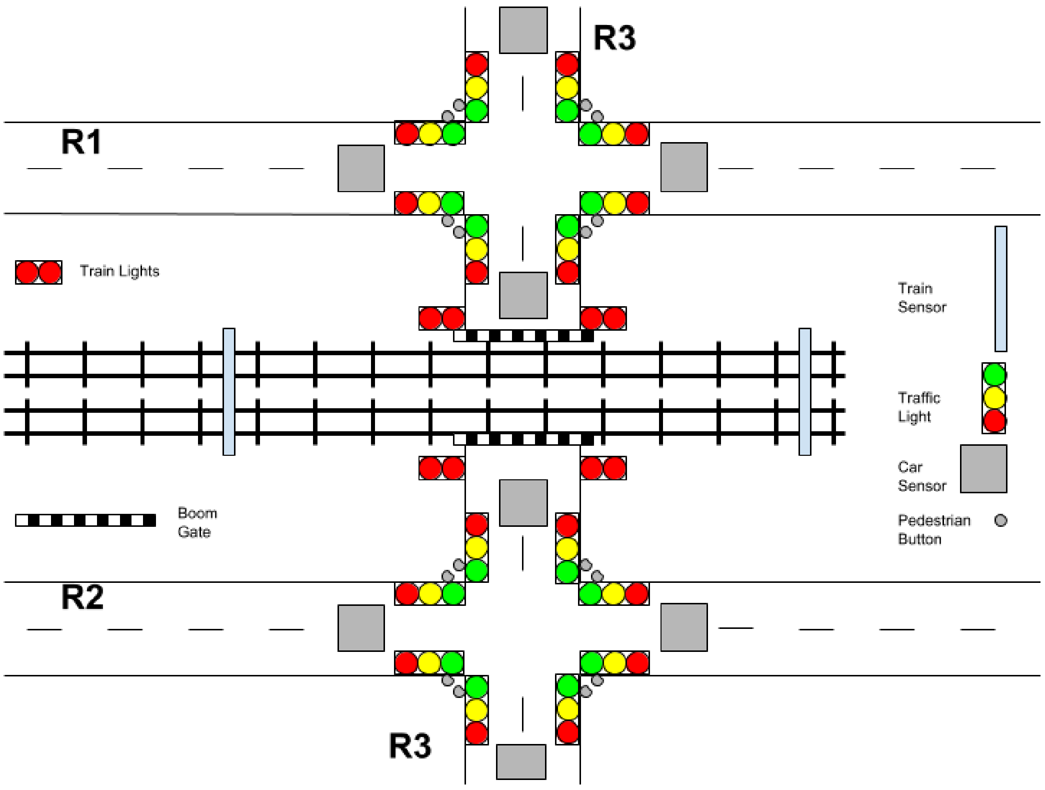


Figure 1: Intersection diagram

The road R3 represents the North-South direction, While the two R1 and R2 roads represent an East-West direction. A dual carriage railway crossing (X1) cuts through the middle of the two traffic lights. Each road can carry two lanes of bidirectional traffic. Pedestrians are able to cross roads R1, R2, and R3 at each traffic light. Cars are able to make turns at each traffic intersection, however dedicated turn lights are not at any intersection. Heavy traffic is expected at road R1 and R3.

The railway crossing has lights and boom gates at each section. Heavy train traffic is to be expected during the day, while light train traffic exists after 2200.

The traffic lights receive information from the railway crossing in order to make informed decisions in calculating the current traffic light state. In the case of a boom gate error, the central control point receives an indication.

Each intersection and crossing operates as an individual node that runs continuously, regardless of network connectivity status. If the nodes are connected, they communicate with a central control point. The central control point is able to override the various error states of the nodes, as well as send scheduling information.

The light sequence patterns for both the traffic lights and railway crossing are dictated both scheduling and sensing. Time scheduling is set by the central controller (or by a default value), while sensing is used to ensure the traffic/railway lights are only changing if absolutely needed. Central control is also able to override the state of scheduling of any node at a moment's notice.

4.2 Context Overview

The system's design may be split up to 3 main sections, the two traffic light controller nodes, the train intersection and boom gate controller node, and the central controller node. These can then be further separated into their own QNX processes running on separate hardware.

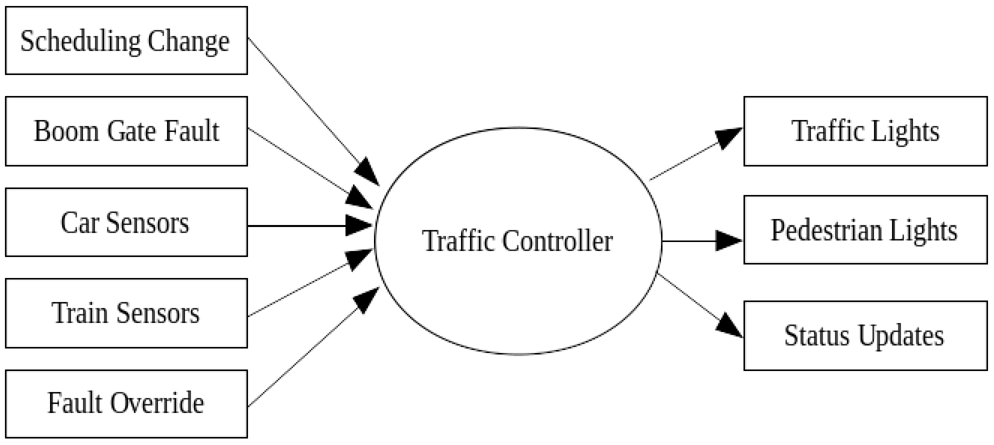


Figure 2: Traffic Controller Context Diagram

The traffic light controller will be responsible for handling the traffic lights themselves, lights for the pedestrians to cross, and the status updates for the central controller sent via native QNX message passing.

Various inputs can alter the operation of the traffic light controller, these include: Scheduling changes sent from the central controller to change from a sensor or time based schedule. Messages sent from the train intersection controller indicating that there is a fault within it's system, and to keep certain traffic lights red to stop traffic from entering for safety. Car sensors to change state for when the node is in sensor scheduling mode. Train sensors to change traffic light state, allowing one side of traffic to flow while the other is blocked from the oncoming train. Finally, the fault override signal sent from the central controller to inform the traffic controller than any boom gate faults have been cleared.

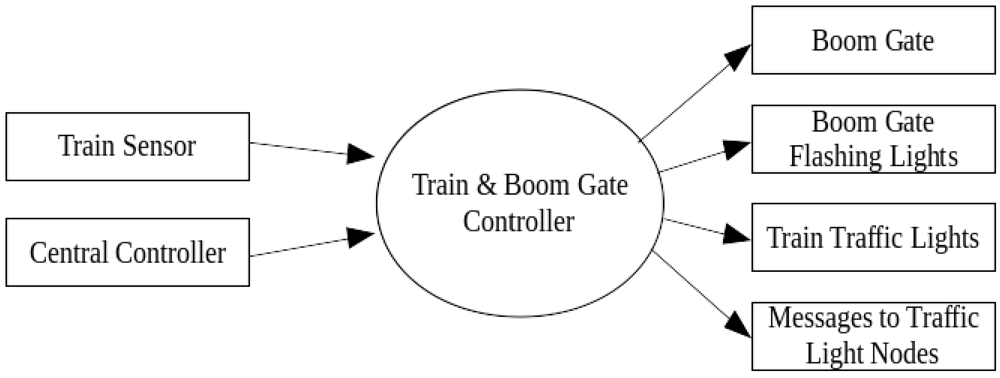

Train Sensor
Central Controller
Train & Boom Gate Controller
Boom Gate
Boom Gate Flashing Lights
Train Traffic Lights
Messages to Traffic Light Nodes

Figure 3: Train Intersection Controller Context Diagram

The train controller is responsible for handling the boom gate closing and opening, flashing the boom gate lights for commuters while a train is oncoming, stopping any trains from passing if there is a major fault via traffic lights specifically for the trains conductors, and sending messages to the traffic light nodes when a train is oncoming.

The inputs to the node will be the oncoming and outgoing train sensors, and messages from the central controller such as for fault overriding.

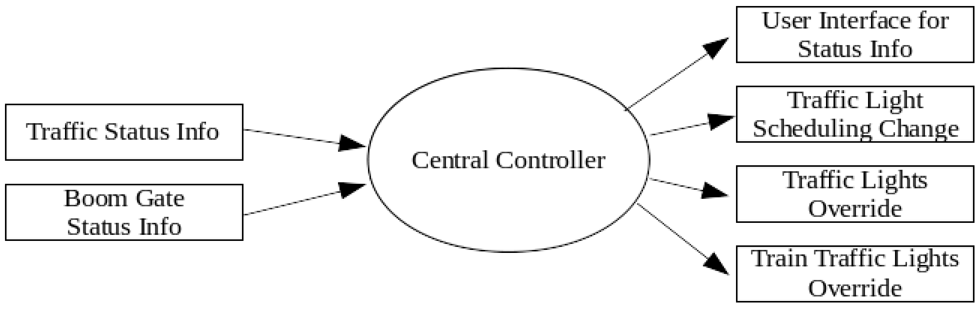

Traffic Status Info
Boom Gate Status Info
Central Controller
User Interface for Status Info
Traffic Light Scheduling Change
Traffic Lights Override
Train Traffic Lights Override

Figure 4: Central Controller Context Diagram

The central controller's main role is to monitor the status of the traffic lights and train intersection. It does this by reading the messages sent from all of the traffic and train nodes within the entire network that are sent every time the nodes change state. From this, the central controller will provide a terminal like user experience to the people in contact with it.

The central controller does not directly control how the nodes within the network operate, but with the traffic light nodes, it can send a message to operate on a sensor, or a timing based schedule. Furthermore, a person operating the central controller is required to override any error states of nodes within the network.

4.3 Use Case Overview

The use case of the traffic light controller is to safely control the flow of traffic, pedestrians and trains through a set of intersections by allowing traffic to pass only when it is safe to do so.

Listed below are a few expected use case scenarios that will be used to describe the usual functionality of the system, as well as some scenarios that may occur in exceptional circumstances, such as a failure in any part of the system or in the event of an accident.

4.4 Use Case Scenarios

4.4.1 Commuter Use Case

Car passes straight through the intersection in any direction. As the car approaches an intersection, the sensor is triggered. If the light is already green the car can pass without any action required. If the light is red the traffic light state machine will react accordingly.

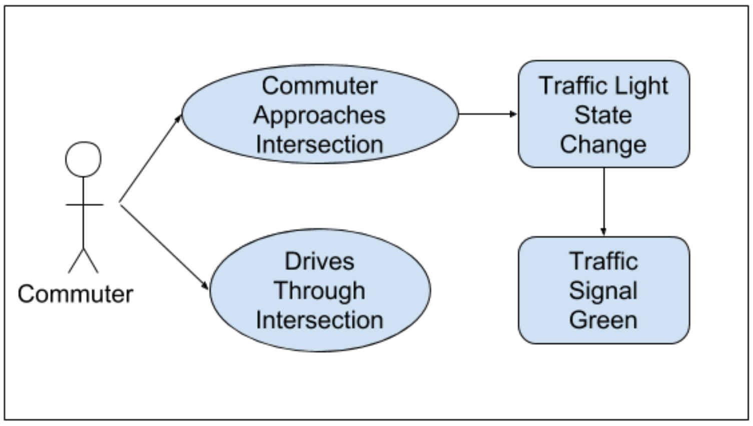

Commuter
Commuter Approaches Intersection
Drives Through Intersection
Traffic Light State Change
Traffic Signal Green

Figure 5: Traffic light commuter use case

4.4.2 Pedestrian Use Case

Pedestrian presses button to cross road, waits for walk light to turn green, then crosses the road. Triggered when the intersection controller receives an input from a pedestrian button. If East-West (EW) pedestrian button is pressed and NS traffic light is green, after the NS traffic light turns yellow, then red, then the EW walk signal will turn green. If NS pedestrian button is pressed and EW traffic light is green, after the EW traffic light turns yellow, then red, then the NS walk signal will turn green.

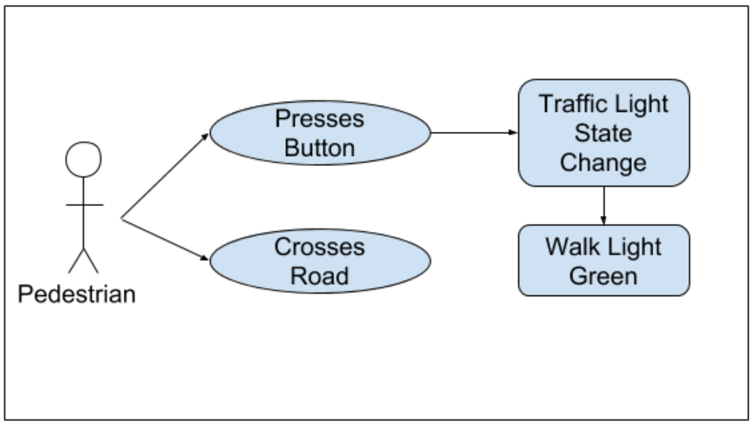


Figure 6: Traffic light pedestrian crossing use case

4.4.3 Train Use Case

Train passes, warning lights turn on, and boom gate activates to prevent traffic from passing in the NS direction. Event is triggered when the train sensor switch is activated and a message is passed to the main controller and the intersection nodes. If the NS traffic light is red, it is prevented from turning green while the train is passing, if the NS traffic light is green when the train sensor switch is triggered it will immediately turn yellow, then red.

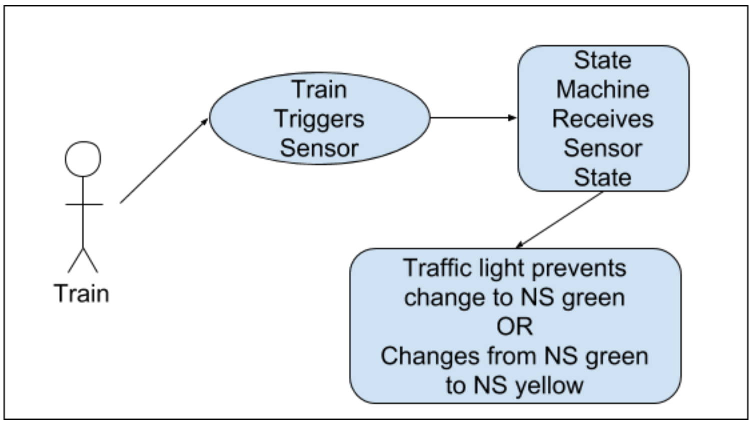

Train
Train Triggers Sensor
State Machine Receives Sensor State
Traffic light prevents change to NS green OR Changes from NS green to NS yellow

Figure 7: Train crossing use case

4.4.4 Train Accident Use Case

A train accident occurs when a train breaks down or crashes in the vicinity of the railway crossing in a way that impacts the ability for cars to cross the railway tracks. A train accident may also mean the failure of the railway controller or message passing, although not necessarily.

If a train accident occurs in a way that does not affect the railway sensing or message passing ability, the system functions as per usual. This would usually mean that the railway sensors would pass a high level to the traffic lights due to the presence of a train. The central control would also be able to send overrides to traffic lights if required.

If the sensor or message passing is impacted, it is up to central control to identify this, and react accordingly by giving the traffic lights the appropriate states to ensure that cars do not approach the railway.

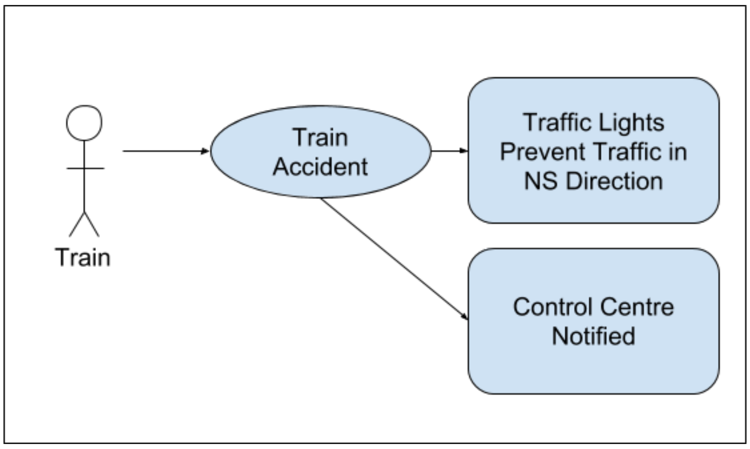


Figure 8: Train accident use case

4.4.5 Traffic Accident Use Case

A traffic accident occurs when a car breaks down or crashes in the vicinity of the traffic light in a way that impacts the ability for cars to cross the traffic lights. A car accident may also mean the failure of the traffic light controller and message passing although not necessarily.

If a traffic accident occurs in a way that does not affect the traffic sensing or message passing ability, the system functions as per usual. This would usually mean that the traffic sensors would pass a high level to the traffic lights due to the presence of traffic. The central control would also be able to send overrides to traffic lights if required.

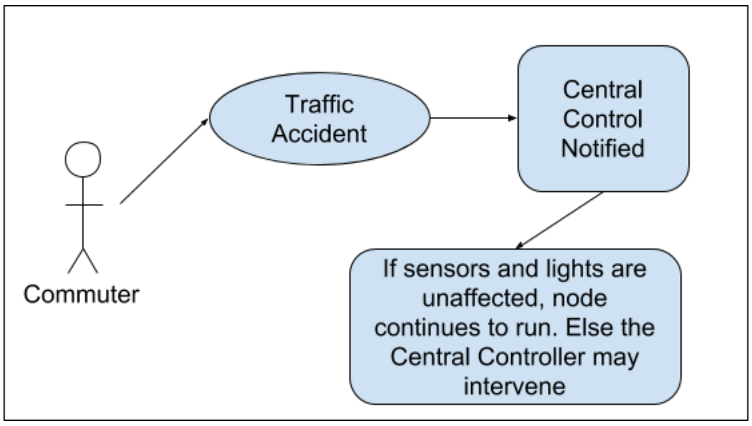

Commuter
Traffic Accident
Central Control Notified
If sensors and lights are unaffected, node continues to run. Else the Central Controller may intervene

Figure 9: Traffic accident use case

4.4.6 Central Control Failure

A central control failure means that the central control no longer can receive or send commands, and information to nodes on the network such as to the traffic lights and railway controllers.

Traffic lights and railway lights/boom gate continue to operate autonomously. Message passing between traffic lights and railway lights/boom gate mean that the system is able to continue to function properly and safely.

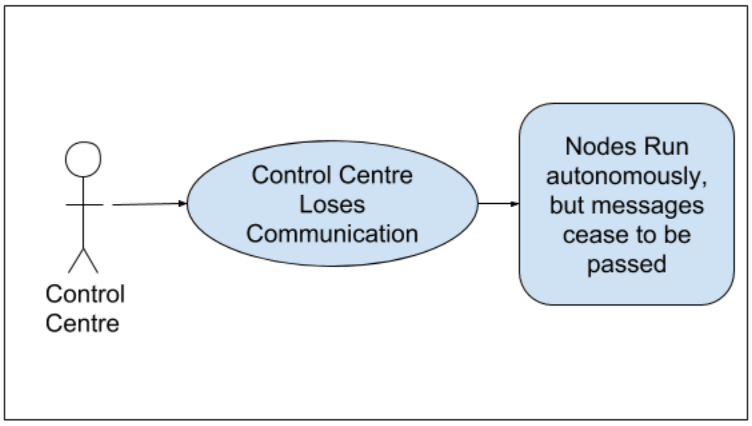

Control Centre
Control Centre Loses Communication
Nodes Run autonomously, but messages cease to be passed

Figure 10: Central control failure use case

4.4.7 Message Passing Failure

In the event that message passing in general fails, and the receiver does not receive a reply to from the recipient, the sender should try sending the message again 4 more times. If a response is still not heard from the recipient, then that means there is a networking fault, and then the local controller should take all precautions it can if need be, and try again later.

4.4.8 Train Controller Node Failure

If there is a train controller failure, the central controller is to use message passing to automatically set adjacent traffic lights to red in order to stop traffic from passing the railway and road intersection. The central controller may then override the error state of the traffic lights manually once everything has been confirmed to be okay.

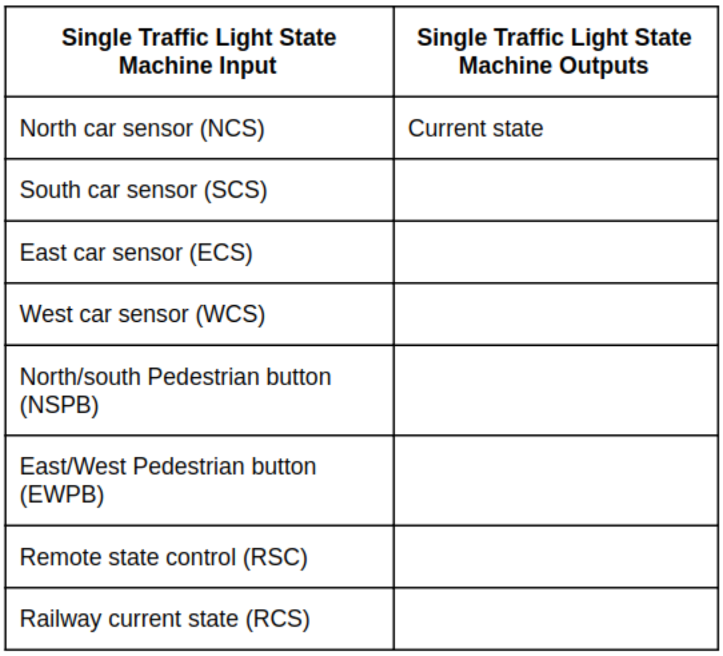
4.4.9 Traffic Controller Node Failure

A failure with the traffic controller node sets the state to an error state where each light at the intersection flashes yellow to signal to commuters about the error. If power to the intersection is lost, then the central controller will not be able to communicate with the traffic controller. Therefore, it will be up to the central controller to identify a loss of power in the node for it to be resolved manually.

4.5 State Overview

4.5.1 Traffic Lights

The following table represents the required inputs and outputs for the traffic light node. These inputs are required in order for the traffic lights to make the correct decision for what to do, enabling them to work autonomously. It includes four car traffic sensor inputs for each side of a intersection, two pedestrian buttons for each direction of pedestrian traffic (ideally this would be four buttons to be perfectly realistic, however due to hardware constraints two button were chosen for the project), a remote state control input for the central control requirements, and the current state of the railway crossing.



Single Traffic Light State Machine Input	Single Traffic Light State Machine Outputs
North car sensor (NCS)	Current state
South car sensor (SCS)	
East car sensor (ECS)	
West car sensor (WCS)	
North/south Pedestrian button (NSPB)	
East/West Pedestrian button (EWPB)	
Remote state control (RSC)	
Railway current state (RCS)	

Figure 11: Traffic light io table

The following diagrams are the traffic light state diagram, and the traffic light state table. It shows the process behind the decision making of the traffic lights, enabling it to operate both autonomously, via remote control.

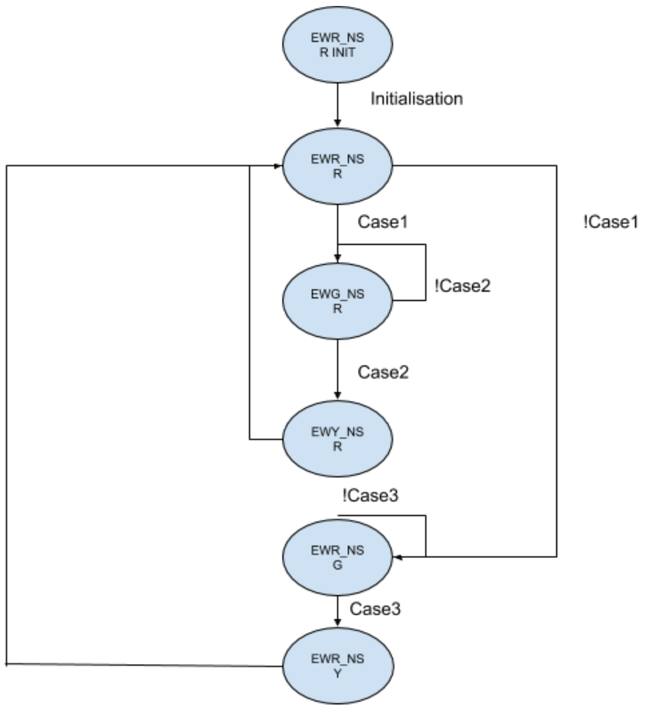
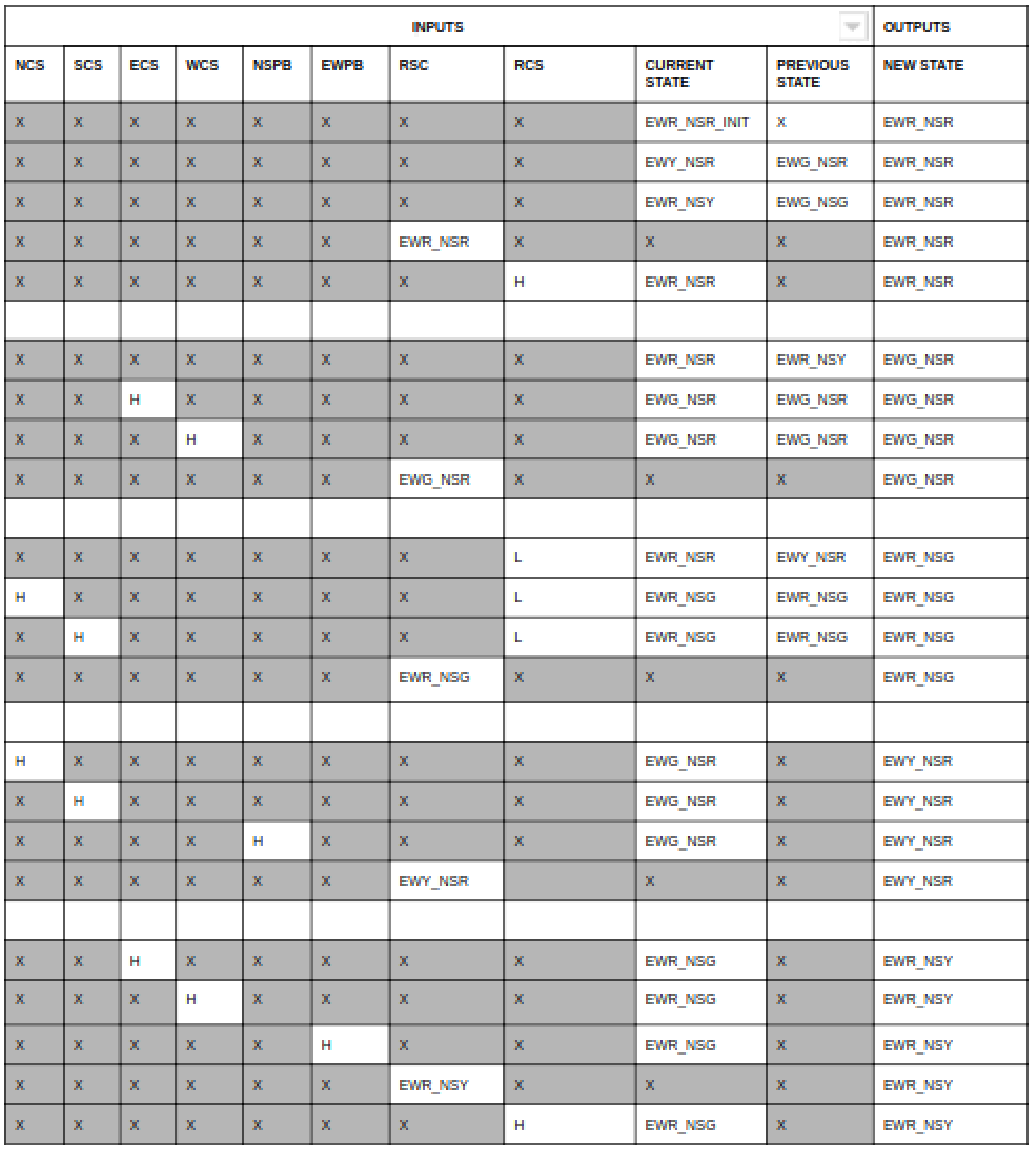


Figure 12: Traffic light state diagram



INPUTS										OUTPUTS
NCS	SCS	ECS	WCS	NSPB	EWPB	RSC	RCS	CURRENT STATE	PREVIOUS STATE	NEW STATE
X	X	X	X	X	X	X	X	EWR_NSR_INIT	X	EWR_NSR
X	X	X	X	X	X	X	X	EWY_NSR	EWG_NSR	EWR_NSR
X	X	X	X	X	X	X	X	EWR_NSY	EWG_NSG	EWR_NSR
X	X	X	X	X	X	EWR_NSR	X	X	X	EWR_NSR
X	X	X	X	X	X	X	H	EWR_NSR	X	EWR_NSR
X	X	X	X	X	X	X	X	EWR_NSR	EWR_NSY	EWG_NSR
X	X	H	X	X	X	X	X	EWG_NSR	EWG_NSR	EWG_NSR
X	X	X	H	X	X	X	X	EWG_NSR	EWG_NSR	EWG_NSR
X	X	X	X	X	X	EWG_NSR	X	X	X	EWG_NSR
X	X	X	X	X	X	X	L	EWR_NSR	EWY_NSR	EWR_NSG
H	X	X	X	X	X	X	L	EWR_NSG	EWR_NSG	EWR_NSG
X	H	X	X	X	X	X	L	EWR_NSG	EWR_NSG	EWR_NSG
X	X	X	X	X	X	EWR_NSG	X	X	X	EWR_NSG
H	X	X	X	X	X	X	X	EWG_NSR	X	EWY_NSR
X	H	X	X	X	X	X	X	EWG_NSR	X	EWY_NSR
X	X	X	X	H	X	X	X	EWG_NSR	X	EWY_NSR
X	X	X	X	X	X	EWY_NSR		X	X	EWY_NSR
X	X	H	X	X	X	X	X	EWR_NSG	X	EWR_NSY
X	X	X	H	X	X	X	X	EWR_NSG	X	EWR_NSY
X	X	X	X	X	H	X	X	EWR_NSG	X	EWR_NSY
X	X	X	X	X	X	EWR_NSY	X	X	X	EWR_NSY
X	X	X	X	X	X	X	H	EWR_NSG	X	EWR_NSY

Figure 13: Traffic light state table

4.5.2 Railway Crossing

The following table represents the required inputs and outputs for the railway crossing node. These inputs are required in order for the railway crossing node to make the correct decision for what to do, enabling it to work autonomously. It includes two train sensor inputs for each side of a crossing, and a remote state control input for the central control requirements.

Single Railway State Machine Input	**Single Railway State Machine Output**
Train 1 sensor	Current state
Train 2 sensor	
Remote state control	

Figure 14: Rail I/O table

The following diagrams are the railway crossing state diagram, and the railway crossing state table. It shows the process behind decision making process of the railway crossing, enabling it to operate both autonomously, via remote control.

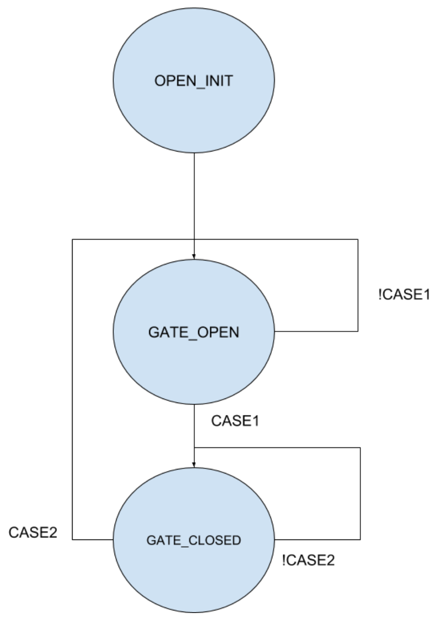
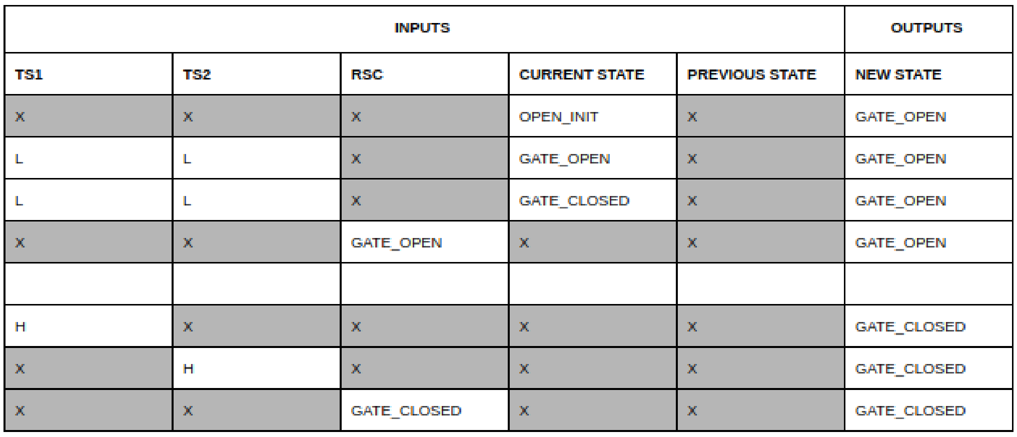


Figure 15: Rail state diagram



INPUTS					OUTPUTS
TS1	TS2	RSC	CURRENT STATE	PREVIOUS STATE	NEW STATE
X	X	X	OPEN_INIT	X	GATE_OPEN
L	L	X	GATE_OPEN	X	GATE_OPEN
L	L	X	GATE_CLOSED	X	GATE_OPEN
X	X	GATE_OPEN	X	X	GATE_OPEN
H	X	X	X	X	GATE_CLOSED
X	H	X	X	X	GATE_CLOSED
X	X	GATE_CLOSED	X	X	GATE_CLOSED

Figure 16: Rail state table

4.6 Task Architecture Diagram

The following diagram describes the task architecture of the system. It provides a broad overview of each task in the system, and which tasks communicate with each other.

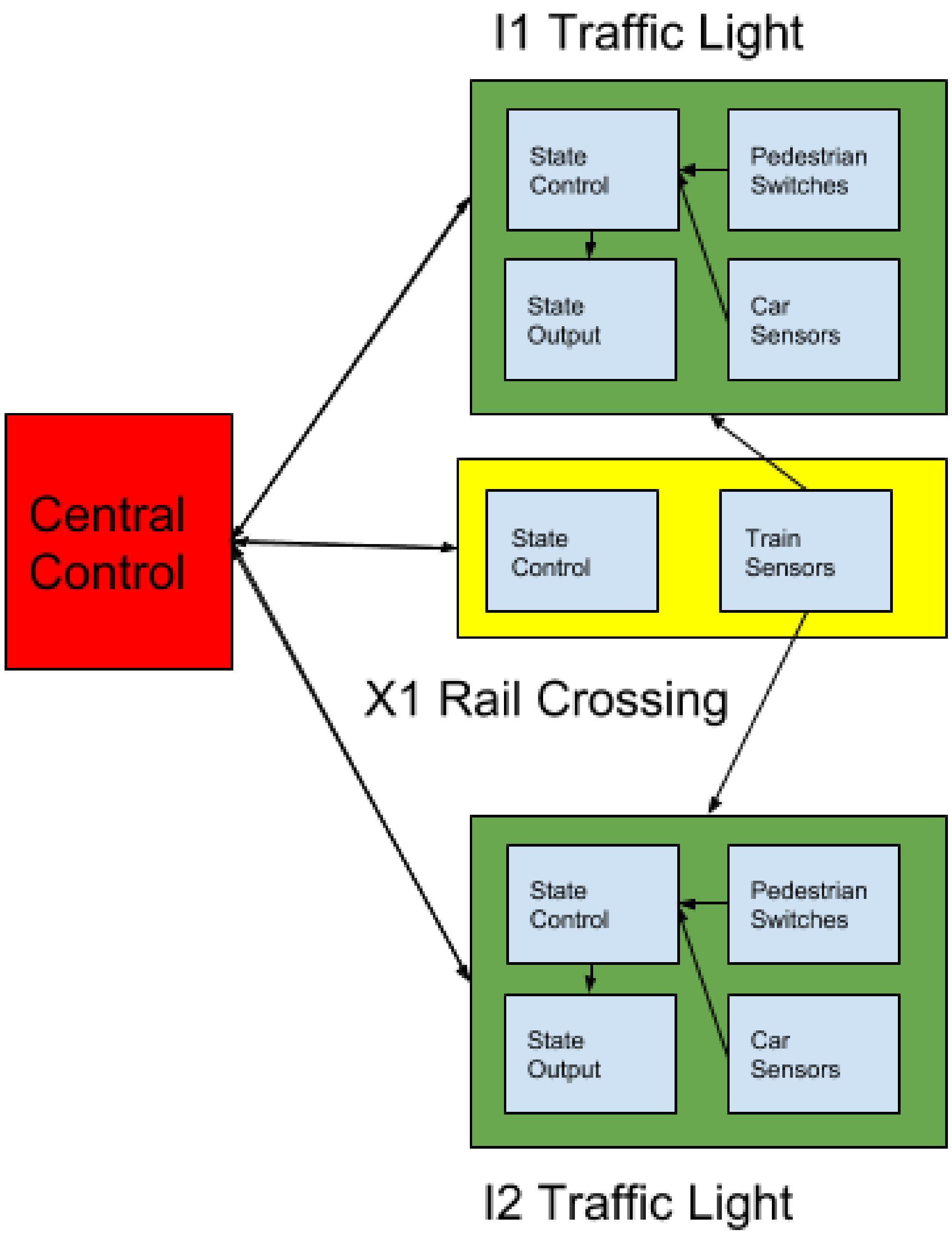


Figure 17: Task architecture diagram

4.7 Messages Passing

4.7.1 Traffic Light Messages

Sensor status for the two pedestrian buttons and the four car sensors are sent to central control as the sensor input status changes, as are Changes in traffic light state. Messages defining the timing schedule of the traffic light are accepted from central control, as well as messages overriding the current traffic light state. Finally, messages from the railway crossing containing the current railway crossing state are received so the traffic lights can alleviate congestion with the coming and going of trains.

4.7.2 Railway Crossing Messages

Sensor status for the two train sensors are sent to central control as the sensor input status changes. Changes in railway state are also sent to central control as the state changes. Messages from central command overriding the current railway crossing state are accepted.

4.7.3 Central Control Messages

All sensor statuses from the traffic lights and railway crossing are accepted, as well as the various current states of these nodes. The central control can also send messages to all nodes overriding their error state.

4.8 Node Overview

The following figures show the hardware block diagrams for the traffic light nodes and railway crossing node. They consist of a series of sensors, buttons, switches, lights, and a DE10-Nano board.

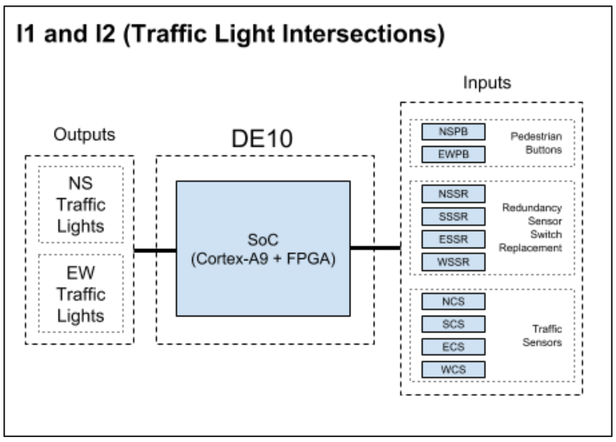


Figure 18: Traffic light block diagram

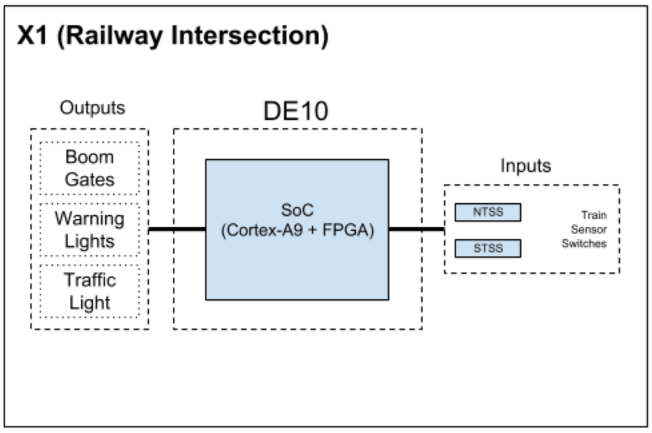


Figure 19: Rail block diagram

4.9 Node I/O

The following figures give a closer look to the inputs and outputs of the FPGA module in the DE10-Nano board for both the traffic light implementation, and the railway crossing implementation.

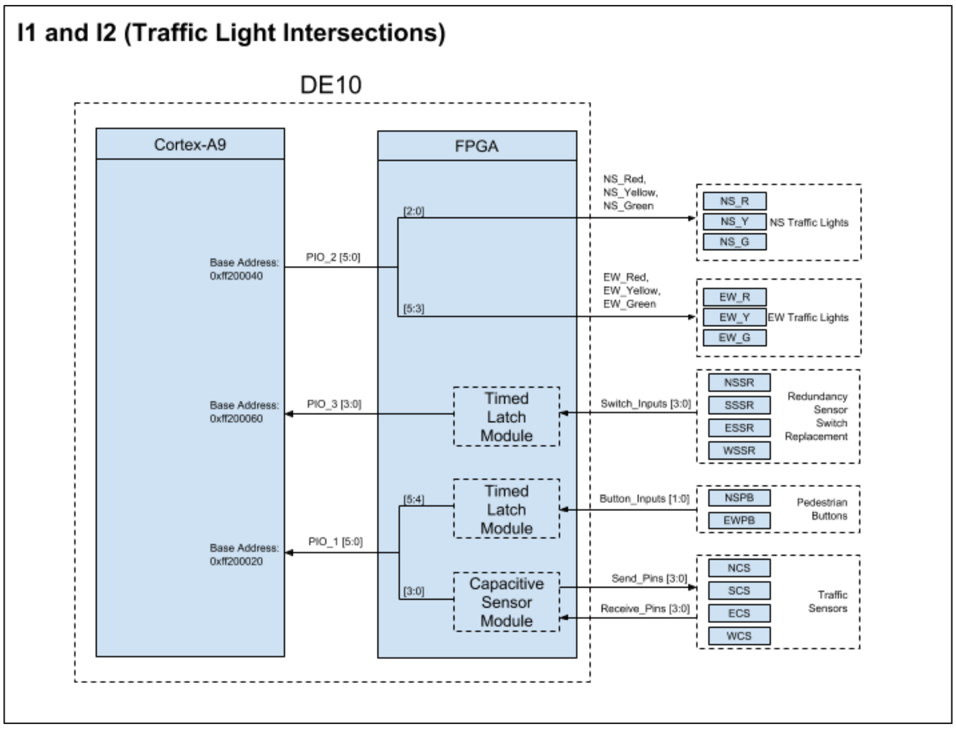
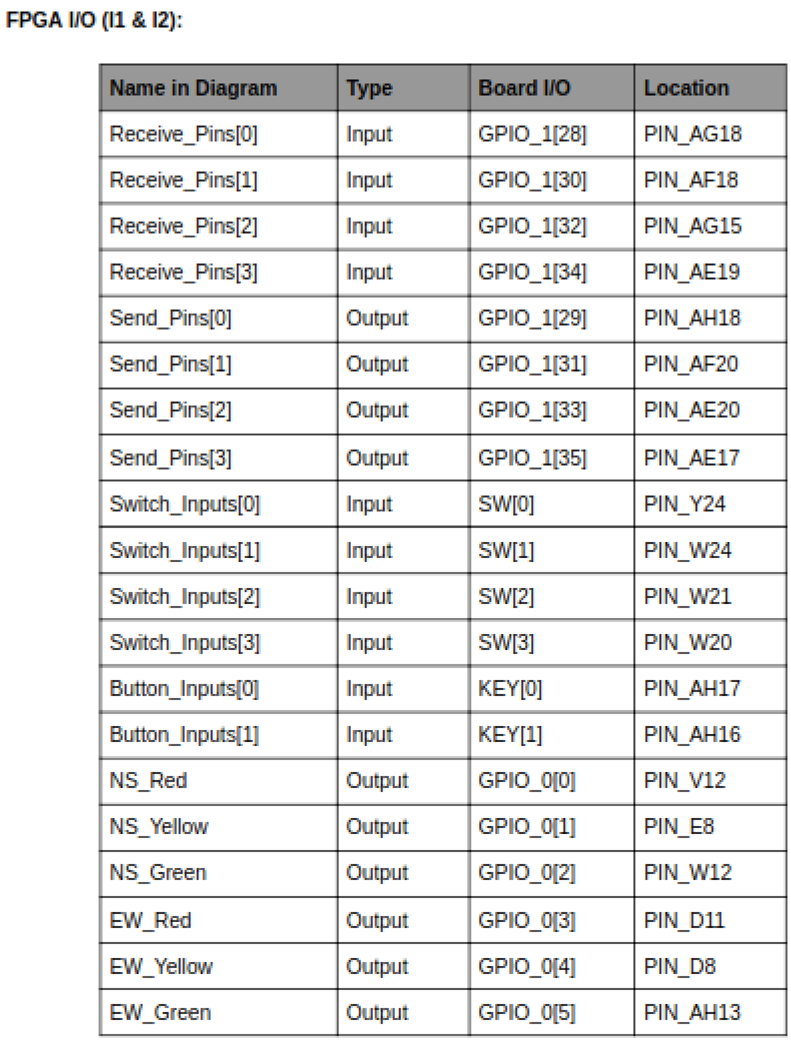


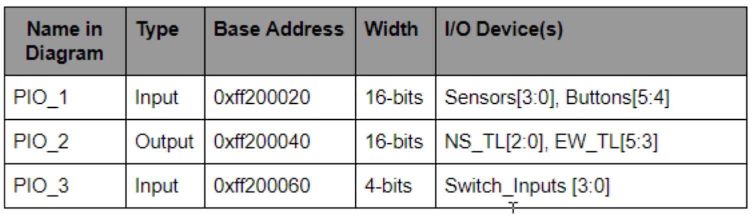
Figure 20: Traffic light I/O diagram



FPGA I/O (I1 & I2):

Name in Diagram	Type	Board I/O	Location
Receive_Pins[0]	Input	GPIO_1[28]	PIN_AG18
Receive_Pins[1]	Input	GPIO_1[30]	PIN_AF18
Receive_Pins[2]	Input	GPIO_1[32]	PIN_AG15
Receive_Pins[3]	Input	GPIO_1[34]	PIN_AE19
Send_Pins[0]	Output	GPIO_1[29]	PIN_AH18
Send_Pins[1]	Output	GPIO_1[31]	PIN_AF20
Send_Pins[2]	Output	GPIO_1[33]	PIN_AE20
Send_Pins[3]	Output	GPIO_1[35]	PIN_AE17
Switch_Inputs[0]	Input	SW[0]	PIN_Y24
Switch_Inputs[1]	Input	SW[1]	PIN_W24
Switch_Inputs[2]	Input	SW[2]	PIN_W21
Switch_Inputs[3]	Input	SW[3]	PIN_W20
Button_Inputs[0]	Input	KEY[0]	PIN_AH17
Button_Inputs[1]	Input	KEY[1]	PIN_AH16
NS_Red	Output	GPIO_0[0]	PIN_V12
NS_Yellow	Output	GPIO_0[1]	PIN_E8
NS_Green	Output	GPIO_0[2]	PIN_W12
EW_Red	Output	GPIO_0[3]	PIN_D11
EW_Yellow	Output	GPIO_0[4]	PIN_D8
EW_Green	Output	GPIO_0[5]	PIN_AH13

Figure 21: Traffic light FPGA I/O table



Name in Diagram	Type	Base Address	Width	I/O Device(s)
PIO_1	Input	0xff200020	16-bits	Sensors[3:0], Buttons[5:4]
PIO_2	Output	0xff200040	16-bits	NS_TL[2:0], EW_TL[5:3]
PIO_3	Input	0xff200060	4-bits	Switch_Inputs [3:0]

Figure 22: Traffic light Cortex-A9 I/O table

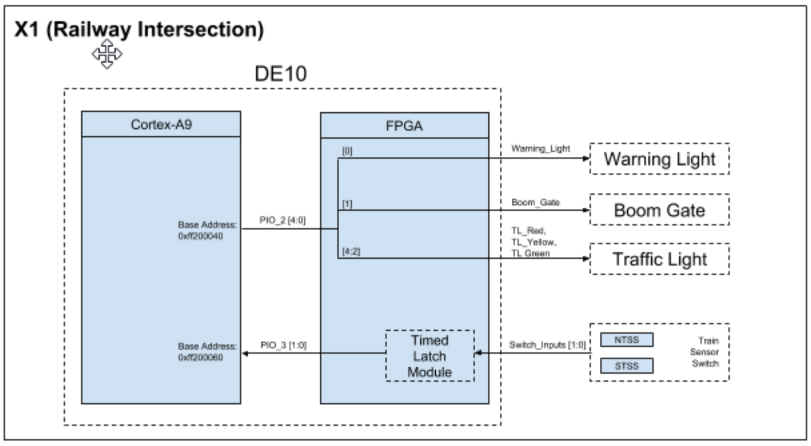
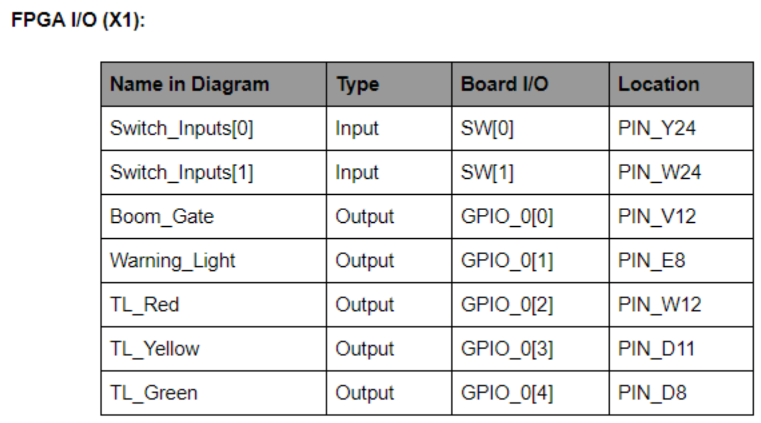


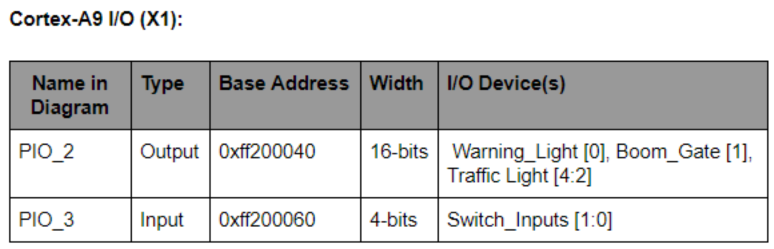
Figure 23: Rail I/O diagram



FPGA I/O (X1):

Name in Diagram	Type	Board I/O	Location
Switch_Inputs[0]	Input	SW[0]	PIN_Y24
Switch_Inputs[1]	Input	SW[1]	PIN_W24
Boom_Gate	Output	GPIO_0[0]	PIN_V12
Warning_Light	Output	GPIO_0[1]	PIN_E8
TL_Red	Output	GPIO_0[2]	PIN_W12
TL_Yellow	Output	GPIO_0[3]	PIN_D11
TL_Green	Output	GPIO_0[4]	PIN_D8

Figure 24: Rail FPGA I/O Table



Cortex-A9 I/O (X1):

Name in Diagram	Type	Base Address	Width	I/O Device(s)
PIO_2	Output	0xff200040	16-bits	Warning_Light [0], Boom_Gate [1], Traffic Light [4:2]
PIO_3	Input	0xff200060	4-bits	Switch_Inputs [1:0]

Figure 25: Rail Cortex-A9 I/O Table

5 Conclusion

We believe this design, implemented, will be capable of directing traffic and pedestrians in the set of traffic intersections described, that the system is safe, efficient and robust enough to run autonomously while allowing for monitoring or intervention in the event of an error.

Due to the use of distributed processing and the fact that each node can run autonomously under normal circumstances, the system could be scaled well to include many more traffic intersections within the network.