Software demonstrator: “Extended Double Pelican Crossing” traffic light system

Project Report

Project Authors:
Chioma Jessica Nkem-Eze
Oluwadamiloju Abraham Yusuf

TALTECH 2022

Table of Content

.

1. Vision Statement

1.1 Introduction

The system being specified is an “Extended Double Pelican Crossing” traffic light system. A traffic light system is a signalling system that establishes priority rules at road intersections and pedestrian crossings in order to control traffic flow and ensure the safety of all users. This traffic light system has the following features:

- Support for a dual carriageway (A divided highway by a central reservation).
- Two interrelated pelican crossings: The pelican crossing is a type of pedestrian crossing that is operated by pedestrian call buttons, with the pedestrian light signal located across the road from the pedestrian.
- Emergency pass-through: A sensing and signalling system that disrupts normal operations to allow emergency vehicles to pass safely.

1.2 Problem statement

The goal of this project is to create a traffic light system that takes into account all relevant real-time data in order to provide smoother-flowing traffic, prioritize emergency vehicles when necessary, and make it easier for pedestrians to cross a dual carriageway by utilizing the pelican crossing system.

1.3 Users of the system

Two categories of users will utilize the system. The pedestrians attempting to cross the highway are the first in this category. The second category consists of motorists who want smoother traffic flow. The term "motorist" refers to all forms of road vehicles. Emergency vehicles, for example, are a significant class of motorists that require the system to expedite their circulation.

1.4 Behavior of the system

By default, the system signals vehicles to travel freely. This signal state will be maintained until a pedestrian on either side of the highway presses the call button. As a result, the system alerts vehicles to slow down and halt for pedestrians who wish to cross and gives them enough time to do so. After this period has passed and all motorists have come to a complete stop, the system alerts pedestrians that it is safe to cross. After this time elapses and all motorists have stopped, the system signals to pedestrians that it is safe to cross.

This behaviour deviates slightly when emergency vehicles seek to pass through the crossing. If the pedestrian presses the call button when the system has already received information about an oncoming emergency vehicle. In that case, the system alerts the pedestrian with a bespoke light indication that the pedestrian must wait because an emergency vehicle is approaching.

1.5 Controlled Objects

The system's controller will control a pair of signal heads for vehicles with lights red, yellow, and green, as well as a pair of signal heads for pedestrians with lights red, green, and yellow. Each lane is also equipped with the following input devices:

- Passive Infrared (PIR) sensors which detect infrared signals from emergency vehicles within a specific range.
- Call buttons tell the controller that pedestrians are present and ready to cross.
- An external timer sends a pulse to the controller to facilitate timing synchronisation.

1.6 Safety Analysis

Given the system's actors, it is critical to underline that system failure can result in several fatalities and significant damage to the vehicles involved in a traffic accident. As a result, the system is classified as critical and essential, with a criticality level of "Catastrophic [A]."

2. System Requirements and Development

2.1 High-Level Requirements

The requirements specified below give an overview of the system's functionalities. There are presently 17 functional requirements and six non-functional requirements.

2.1.1 Functional Requirements

- HLR1 - Upon resumption of power to the controller, no signals SHALL be displayed for a time not less than x seconds nor greater than y seconds before normal operations begin.
    - HLR1.1 - If the system has already been turned off for more than x seconds, the system shall skip the startup delay. Otherwise, the system shall start operations only after a fixed delay.
    - HLR1.2 - The controller shall resume operation at the "Vehicles Passing Ready" phase. That is a yellow light signal for the vehicle light set and a red light signal for the pedestrian light set.

- HLR2 - The traffic lights for both carriageways shall remain in one default configuration until a call button on either side of the carriageway is pressed. This configuration is green for the vehicle light set and red for the pedestrian light set.

- HLR3 - When the call button on either carriageway is pressed, the system shall initiate the sequence for light transition to green for pedestrians and red for motorists.
    - HLR3.1 - When the call button on either carriageway is pressed, the system shall signal the pedestrian to cross in no longer than X seconds.
    - HLR3.2 - The system shall automatically trigger a button press on the other carriageway after X seconds when the call button on either carriageway is pressed.
    - HLR3.3 - When the call button on either carriageway is pressed and the light sequence has initiated, the system shall not service any new requests.

- HLR4 - When the system receives an approach signal from an emergency vehicle on one carriageway, the system shall initiate the preemption process.
    - HLR4.1 - When the system receives an approach signal from an emergency vehicle on one carriageway, the pedestrian traffic light shall be a red colour with flashing effects for that carriageway.
    - HLR4.2 - When the system receives an approach signal from an emergency vehicle on one carriageway, the controller shall transition from its current state at the end of its duration to the emergency vehicle approaching state. The exception to this is when the pedestrian traffic light is green.

    - HLR4.3 - After the emergency vehicle has passed through, the traffic light state shall transition from the emergency vehicle approaching state to the vehicle slow down state.
    - HLR4.4 - When the system receives an approach signal from an emergency vehicle on one carriageway, all button presses are ignored until after the emergency vehicle has passed through.

- HLR5 - The controller shall switch the traffic lights based on the standard lights regime: Green - Yellow - Red and vice versa. The yellow lights must always be between the green and red lights transitions.
    - HLR5.1 - The vehicle and pedestrian light set on one carriageway must never be green at the same time.
    - HLR5.2 - When the light switching occurs, each signal head must display no more than one light signal at a time.
    - HLR5.3 - x seconds after the pedestrian light set has been switched to red; the controller shall switch the vehicle light set to green.

2.1.2 Non-Functional Requirements

- HLR6 - The system's scope shall consist of a single controller that controls a pair of signal heads for motorists and a pair of signal heads for pedestrians on both carriageways.

- HLR7 - The system shall detect infrared signals through passive infrared sensors within a minimum range of X meters.

- HLR8 - The system shall perform commands issued to it in no more than X milliseconds.

- HLR9 - The system shall aim for 99.999% availability yearly i.e the system can only be down for 6 minutes per year.

- HLR10 - The system shall recover normal operations after a system failure in no more than X seconds.

- HLR11 - The system shall initiate fail-safe operation during system failures in no more than X seconds.

2.2 Low-Level Requirements

The requirements specified below refine the high-level requirements given above and provide a prospective look at how system components will be implemented.

- LLR1 - This operation gets the downtime (in milliseconds) from the external system, DowntimeTracker. If the downtime is less than the configured startup delay threshold, it waits for the configured startup delay before changing the state to

'vehiclesPassingReady'. Otherwise, if the downtime exceeds the configured startup delay threshold, it changes the state immediately to 'vehiclesPassingReady'. This operation returns no value.

- LLR2 - This function accepts the carriageway on which the state is to be changed and a string indicating the state as a parameter. It sets the callButtonReady flag to false, indicating that any button presses should be ignored for the time being. It also retrieves the state's properties (i.e. colour and flashing states of the vehicle and pedestrian lights, duration of the state, next state, next action) from the state config object and then invokes the carriageway display function to display the appropriate lights. After this, it delays for the time specified in the state's duration attribute and then gets the emergencySignalReceived flag from the carriageway class. Suppose the flag is true, and the state is not 'pedestriansCrossing' and 'pedestriansCrossingAlmostDone'. It performs the following operations:

    - Sets the emergencySignalReceived flag to false,
    - Sets the nextState property to 'emergencyVehicleApproaching',
    - and Sets the nextAction property to false

  If the nextState property exists, it calls itself with the carriageway argument and the nextState. However, if the nextAction property exists, indicating that the "vehiclesPassing" state duration is complete, the operation vehiclePassingDurationComplete is called. The operation returns no value.

- LL3 - This operation accepts the carriageway on which the call button press is to be processed as a parameter. It gets the callButonPressed flag from the carriageway class. If the callButtonPressed flag is true, it performs the following operations:

    - It changes the callButtonPressed flag to false
    - Because the carriageway whose call button was pressed is passed as an argument, it gets a reference to the 'other' carriageway.
    - It invokes the setCarriagewayState function with the state 'vehiclesPassingSlowDown' for the carriageway passed in.
    - It calls the setCarriagewayState function for the 'other' carriageway after a defined carriagewayWaitPeriod.

  If the callButtonPressed flag is false, indicating that there are no button presses to process, no action is taken. This operation is only performed when the state, vehiclePassing's wait duration has expired, or after the system has received a call button press and the callButtonReady flag is set to true. This operation returns no value.

- LL4 - This operation accepts the carriageway on which the call button press is to be processed as a parameter. It sets the callButtonReady flag to true, which indicates that the

minimum wait period for vehicle passing has elapsed. It also invokes the processCallButtonPress function passing in a reference to the carriageway on which the call button was pressed. This operation returns no value.

- LL5 - When invoked, this operation sets the emergencySignalReceived flag to true. It returns no value. It is important to note that when the controller changes the carriageway state, it checks to see if the emergencySignalReceived flag is true. If it is, it changes the state to the 'emergencyVehicleApproachingState'.

- LL6 - This operation accepts the colour and flashing state configuration of vehicles and pedestrian lights on a specific carriageway. It then changes the current colour and flashing effect state to the new one. It returns no value.

- LL7 - When invoked, this operation sets the callButtonPressed flag to true. If the callButtonReady flag is true, it proceeds to call the processCallButtonPress function on the controller. It passes a reference to itself as an argument. If the callButtonReady flag is false, it does nothing. This operation returns no value.

2.3 Traceability Matrix

The table below depicts the link between functional high-level and low-level requirements. The low-level requirements do not refine the non-functional requirements since they are too abstract for the current implementation.

	LLR1	LLR2	LLR3	LLR4	LLR5	LLR6	LLR7
HLR1	X						
HLR1.1	X						
HLR1.2	X						
HLR2		X	X	X			
HLR3				X			X
HLR3.1							X
HLR3.2			X				
HLR3.3		X					
HLR4					X		
HLR4.1		X				X	
HLR4.2		X			X		
HLR4.3		X					
HLR4.4		X			X		

HLR5		X					
HLR5.1		X					
HLR5.2		X				X	
HLR5.3		X					

2.4 Implementation Summary

We constructed a console app for the implementation. It is possible to submit inputs and see the terminal display change as the system responds to those inputs. Because of the team's expertise with JavaScript, it was chosen as the language of choice. We had to tweak some of the operations in our classes to accommodate the language feature as well as the console feature limitation. However, the system prerequisites are met, and the following features may be manually tested:

- Startup sequence with delay condition handling
- Receive and respond to button press on C1 and C2
- Receive and respond to emergency signal on C1 and C2
- Trigger button press on ‘other’ carriageway after a delay

To test the functionalities on your own computer, please do the following:

- cd into the code directory in the lab1 directory,
- run `npm install` in the terminal,
- run `npm start` and follow instructions,
- to run tests, run `npm test`. Note: The tests are time based, and so, it might take some time verifying.
- After exiting from the test interface, the coverage analysis of the test is shown.
- Note: If the installation does not work, please ensure you have at least node v16 installed on your computer.

3. System Verification

3.1 Test Cases Specification

We used informal analysis for the majority of the HLRs and LLRs due to time and implementation restrictions. We have simply chosen a few requirements to test. These requirements were chosen because they are crucial to the system.

Test Case ID	TC-001
Description	Verify that the controller transitions to the “vehiclesPassingSlowDown” state on C1 after button press on C1
Requirement Reference	HLR3, LLR7
Input Description	C1, a carriageway instance. A constant, minimumWaitTime (10s). This defines the minimum amount of time for vehicles to move freely.
Preconditions	The controller is active, and a button press has been recorded on C1
Expected Result	The system transitions from the “vehiclesPassing” state to the “vehiclesPassingSlowDown” state on C1
Success Criteria	Vehicles Lights: GREEN (FLASHING), Pedestrians Lights: RED

Test Case ID	TC-002
Description	Verify that the controller transitions to the “vehiclesPassingSlowDown” state on C2 after button press on C1
Requirement Reference	HLR3.2
Input Description	C1 and C2, instances of a carriageway. The constants, minimumWaitTime (10s) and carriagewayWaitDuration (8s). The minimumWaitTime defines the minimum amount of time for vehicles to move freely. The carriagewayWaitDuration specifies the time elapsed between the button push on C1 and the following trigger on C2.

Preconditions	The controller is active, and a button press has been recorded on C1
Expected	The system transitions from the “vehiclesPassing” state to the “vehiclesPassingSlowDown” state on C2.
Success Criteria	Vehicles Lights: GREEN (FLASHING), Pedestrians Lights: RED on C2

Test Case ID	TC-003
Description	Verifies that the controller transitions from the "pedestrianCrossingDone" state to the “vehiclesPassing” state in no more than 9 seconds
Requirement Reference	HLR5.1
Input Description	C1, a carriageway instance. The constants, pedestriansCrossingDoneDuration (3s), and vehiclesPassingReadyDuration (5s). Both constants state how long their respective states should last.
Preconditions	The carriageway state on C1 is set to ‘pedestriansCrossingDone’.
Expected	The system transitions to the “vehiclesPassing” state in no more that 9 seconds.
Success Criteria	Vehicles Lights: GREEN, Pedestrians Lights: RED

Test Case ID	TC-004
Description	Verifies that the system delay for 10 seconds before transitioning to the “vehiclesPassingReady” state if the downtime duration is less that the startUpDelayThreshold (10s).
Requirement Reference	HLR1.1, LLR1
Input Description	C1, a carriageway instance. A constant, startUpDelayDuration (10s). This defines the amount of time the controller has to delay if the condition is met.

Preconditions	The controller is started
Expected	The system delays for 10 seconds before transitioning to the “vehiclesPassingReady” state
Success Criteria	Vehicles Lights: YELLOW, Pedestrians Lights: RED

3.2 Test Procedures

This project's test procedures are in the form of automated tests utilizing the Mocha javascript test framework. Each of the processes corresponds to one of the test cases stated above. Please see Section 2.4 for installation instructions on how to conduct the tests.

The tests are divided into two main blocks as shown below; The first block starts up the controller and initiates a button press on each test run within it.

```
describe('With Startup Sequence', function () {

  beforeEach(async function () {
    controller.startUpComplete();

    await new Promise(res => setTimeout(res, vehiclesPassingReadyDuration));
    controller.onCarriagewayInput({name: '1'});
  });

  describe('TC-001', function () {
    it('should transition to the “vehiclesPassingSlowDown” state on C1 after button press on C1', async function () {
      await new Promise(res => setTimeout(res, minimumWaitTime));
      const display = await controller.getContentForDisplay(controller.c1);

      assert(display.includes(`${chalk.bold('C1')} Vehicles Lights:            ${chalk['green']('GREEN (FLASHING)')}`));
      assert(display.includes(`${chalk.bold('C1')} Pedestrians Lights:         ${chalk['red']('RED')}`));
    });
  });

  describe('TC-002', function () {
    it('should transition to the “vehiclesPassingSlowDown” state on C2 after button press on C1', async function () {
      const totalDuration = minimumWaitTime + carriagewayWaitDuration + codeLagAdjustment;

      await new Promise(res => setTimeout(res, totalDuration));
      const display = await controller.getContentForDisplay(controller.c2);

      assert(display.includes(`${chalk.bold('C2')} Vehicles Lights:            ${chalk['green']('GREEN (FLASHING)')}`));
      assert(display.includes(`${chalk.bold('C2')} Pedestrians Lights:         ${chalk['red']('RED')}`));
    });
  });

});
```

```
describe('Without startup sequence', function() {
  describe('TC-003', function () {

    before(async function () {
      controller.setCarriagewayState(controller.c1, "pedestriansCrossingDone");
    });

    it('should transition from the "pedestrianCrossingDone" state to the “vehiclesPassing” state in no more than 9 seconds', async function () {
      const totalDuration = pedestriansCrossingDoneDuration + vehiclesPassingReadyDuration + codeLagAdjustment;

      await new Promise(res => setTimeout(res, totalDuration));
      const display = await controller.getContentForDisplay(controller.c1);

      assert(display.includes(`${chalk.bold('C1')} Vehicles Lights:            ${chalk['green']('GREEN')}`));
      assert(display.includes(`${chalk.bold('C1')} Pedestrians Lights:         ${chalk['red']('RED')}`));
    });
  });

  describe('TC-004', function () {
    before(function () {
      controller.start();
    });

    it('should delay for 3 seconds before transitioning to the “vehiclesPassingReady” state', async function () {
      await new Promise(res => setTimeout(res, startUpDelayDuration));      You, 5 minutes ago • Readjusted test cases …
      const display = await controller.getContentForDisplay(controller.c1);

      assert(display.includes(`${chalk.bold('C1')} Vehicles Lights:            ${chalk['yellow']('YELLOW')}`));
      assert(display.includes(`${chalk.bold('C1')} Pedestrians Lights:         ${chalk['red']('RED')}`));
    });
  });
})
```

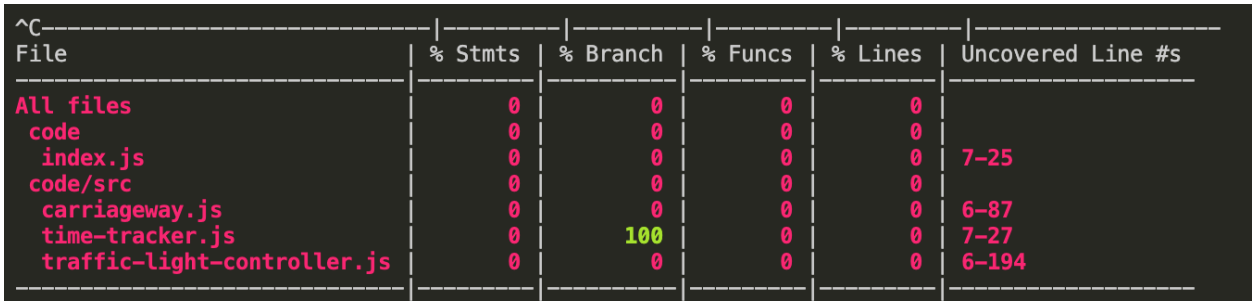
The second block, as indicated above, requires no operations to be performed prior to each test. However, the tests themselves have prerequisites that must be satisfied. The following is a brief summary of how each of these tests implements the test cases:

- TC-001 implements the test case TC-001. It verifies that the controller transitions to the “vehiclesPassingSlowDown” state on C1 after button press on C1. It does this by starting the controller and initiating a button press. It sets a delay for the minimumWaitTime, and then gets the current state for C1 at the time. It asserts that the state meets the success criteria
- TC-002 implements the test case TC-002. It verifies that the controller transitions to the “vehiclesPassingSlowDown” state on C2 after button press on C1. It does this by starting the controller and initiating a button press. It sets a delay for the minimumWaitTime and carriagewayWaitDuration, and then gets the current state for C2 at the time. It asserts that the state meets the success criteria
- TC-003 implements the test case TC-003. It verifies that the controller transitions from the "pedestrianCrossingDone" state to the “vehiclesPassing” state in no more than 9 seconds. It does this by setting the initial state to “pedestriansCrossingDone”. Then it sets a delay for the total duration of the successive states. After the timer elapses, it gets the state for C1 at the time. It asserts that the state meets the success criteria

- TC-004 implements the test case TC-004. It verifies that the system delay for 10 seconds before transitioning to the “vehiclesPassingReady” state if the downtime duration is less that the startUpDelayThreshold (10s). It does this by starting the controller, and then setting a delay for startupDelayDuration. After the timer elapses, it asserts that the state meets the success criteria.

3.3 Tests Report and Coverage Analysis

All of the test procedures are correctly passing with their respective success criterias met. For the coverage report, we were not able to get adequate coverage between the tests and the code, and we are not sure why. Due to time constraints, we have settled for display the coverage table that we have derived.



```
^C------------------------------|---------|----------|---------|---------|-------------------
File                           | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------------------|---------|----------|---------|---------|-------------------
All files                      |       0 |        0 |       0 |       0 |
 code                          |       0 |        0 |       0 |       0 |
  index.js                     |       0 |        0 |       0 |       0 | 7-25
 code/src                      |       0 |        0 |       0 |       0 |
  carriageway.js               |       0 |        0 |       0 |       0 | 6-87
  time-tracker.js              |       0 |      100 |       0 |       0 | 7-27
  traffic-light-controller.js  |       0 |        0 |       0 |       0 | 6-194
-------------------------------|---------|----------|---------|---------|-------------------
```