Light Rail System

Test Plan IEEE 829

Version <1.0>

Table of Contents

1. INTRODUCTION 4
   1.1 PURPOSE 4
   1.2 SCOPE 4
   1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 4
   1.4 REFERENCES 5
   1.5 OVERVIEW 5
2. QUALITY MANAGEMENT 5
   2.1 RESOURCES, ROLES, AND RESPONSIBILITIES 5
   2.1.1 RESOURCES 5
   2.1.2 ROLES AND RESPONSIBILITIES 6
   2.2 SCHEDULES 6
   2.3 PROBLEM REPORTING 7
   2.4 DEPENDENCIES 7
   2.4.1 SOFTWARE DEPENDENCIES 7
   2.4.2 HARDWARE DEPENDENCIES 7
   2.5 RISKS AND ASSUMPTIONS 7
   2.5.1 RISKS 7
   2.5.2 ASSUMPTIONS 8
3. TESTING STRATEGY 8
   3.1 UNIT TESTING 8
      3.1.1 CTC 8
      3.1.2 Track Controller Software 8
      3.1.3 Track Controller Hardware 8
      3.1.4 Track Model 8
      3.1.5 Train Model 8
      3.1.6 Train Controller 9
   3.2 System Testing 9
4. Test Procedure 9

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to provide a comprehensive plan for the development and testing of the renovated Centralized Traffic Control Center (CTC) and Signaling System for the Light Rail Transit System. This plan outlines the testing and quality assurance activities that will be carried out during the development of the system and provides a framework for ensuring that the system is reliable, efficient, and meets the requirements of the stakeholders.

1.2 SCOPE

The scope of this plan is to define the testing and quality assurance approach for the renovated CTC and Signaling System, which consists of six modules: CTC office, software wayside controller, hardware wayside controller, track model, train model, and train controller. The system will simulate transit of the Light Rail Transit system, dispatching trains from Brookline to PNC Park in Pittsburgh, Pennsylvania. The system will be demonstrated on a commercial computer supporting Windows 10 OS and presented to the president of the Pittsburgh Regional Transit's system—the Port Authority of Allegheny County (PAAC)—for adoption and use. The system is also expected to be used by other stakeholders, such as the Transportation Security Administration (TSA) and the National Transportation Safety Board (NTSB).

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Light Rail Transit System	A public transportation system that operates on a dedicated track and uses electric rail vehicles to transport passengers.
Signaling System	A system that controls train movements and ensures safe and efficient operation of the transit system.
LRT	Light Rail Transit
TSA	Transportation Security Administration
IEEE Standard	International Electric Electronic Engineering Standards 1016
NTSB	National Transportation Safety Board
PAAC	Port Authority of Allegheny County
Stakeholders	Any person with an interest in the system who is not a developer of the system.
OS	Operating System
IDE	Integrated Development Environment
User	Person who wants to use the system
User Interface	An interface that the system contacts with the user of the system. It gets all the information needed for its running, from users to the system.
Centralized Traffic Control Center (CTC)	From the office dispatchers control the transit system; Schedule, monitor and route trains, close and open track sections for maintenance, set authority of trains, show current state of entire transit system
Wayside Controller	Controls the track; Communicates with CTC office to control switching of the track, railway crossings, detect broken rails or presence of trains and reports back state of the system to the CTC. Runs a PLC program written by the user

Track Model	Represents the transit system track layout; Determines allowable directions of travel, branching and speed limits, enable track circuits for presence detection and failure detection
Train Model	Models train movement using Newton's laws assuming the train as a point mass and accounting for the terrain of the track; Shows information regarding state of the train and should detect failures
Train Controller	Controls the train; Regulates speed of the train to the setpoint while not exceeding the speed limit or authority allowed by the system, operates the train according to input from a Transit operator, and monitors faults which are acted upon in a safe manner

1.4 REFERENCES

The following documents and standards are referenced in this plan:

- IEEE 829-2017 Standard for Software and System Test Documentation
- The Light Rail Transit System requirements document

1.5 OVERVIEW

The renovated CTC and Signaling System is designed to provide a more efficient and reliable transportation experience for passengers, while increasing the safety and security of the transit system. The system will consist of six modules, each with its own specific function and set of requirements. The CTC office module will provide centralized control and monitoring of the system, while the software and hardware wayside controllers will monitor and control the trains and trackside equipment. The track model module will simulate the physical characteristics of the track, while the train model and train controller modules will simulate the behavior of the trains. The system will be developed in accordance with the requirements outlined in the Light Rail Transit System requirements document and will undergo a rigorous testing and quality assurance process to ensure that it meets the needs of the stakeholders. This document provides a detailed plan for the testing and quality assurance activities that will be carried out during the development of the system.

2. QUALITY MANAGEMENT

2.1 RESOURCES, ROLES, AND RESPONSIBILITIES

GitHub will be used for issue tracking. Each module must push changes to their branch frequently so other members can run tests with the most up to date code.

2.1.1 RESOURCES

- *GitHub:* A web-based platform for version control and collaboration. This is where the repository of the development team's code base can be found.
- *Testing environment:* A set of hardware and software resources required for testing the train simulation system. Unit tests will be written in separate Python scripts and run in an IDE that supports Python.

- *Test data:* A set of inputs and expected outputs for testing the train simulation system. The list of each module's inputs and expected outputs can be found in the project's SRS.
- *Test tools*: A set of software tools required for testing the train simulation system, including automated testing tools and code coverage tools. The Python tool called "unittest" will be used to run each module's unit tests.
- *Documentation*: A set of documents required for the train simulation system, including design documents, user manuals, and test reports.

2.1.2 ROLES AND RESPONSIBILITIES

- *Module Developer*: Responsible for the design, development, and testing of their assigned module, which includes the CTC office, software wayside controller, hardware wayside controller, track model, train model, or train controller. Each module developer is expected to collaborate with other module developers to ensure proper integration between modules, and to provide support to other developers when needed. Module developers must also design and implement unit tests for their own modules.
- *Test Engineer*: Responsible for designing and executing integration and system-level test cases, analyzing test results, and reporting defects. Test engineers for this project are also the module developers and ensure adequate test coverage, as well as identify and troubleshoot issues that arise during testing. Test engineers must also collaborate with other team members to design and implement unit tests for each module.
- *Documentation Specialist*: Responsible for creating and maintaining project documentation, including design documents, installation manuals, and test reports. All team members are expected to provide input and feedback on project documentation, and each module developer must provide documentation for their own module.
- Each module must have its own branch on GitHub, and changes must be pushed frequently to ensure other members have access to the most up-to-date code.
- The project manager will assign issues on GitHub to developers, and developers are responsible for resolving these issues.
- Test cases must be added to the GitHub repository as issues, and members are responsible for assigning these issues to other members when needed.
- Test results and defects must be reported on GitHub as issues, and the test manager is responsible for coordinating with developers to resolve these issues.

2.2 SCHEDULES

The project timeline will follow a series of sprint cycles using the Agile methodology. Sprint cycles will be managed using Jira, with each sprint lasting 2-3 weeks.

Each sprint cycle will consist of the following phases:

1. Planning and design
2. Implementation and coding
3. Testing and quality assurance
4. Documentation and reporting

Each module developer will be responsible for managing their own tasks and deadlines within each sprint cycle. The documentation specialist will be responsible for managing the project documentation and ensuring that it is up to date throughout each sprint cycle. The testing specialist will be responsible for developing and executing the test cases for each module within each sprint cycle.

Each member will oversee the progress of the sprint cycles, ensuring their own module is on track to meet their deadlines and that the project is progressing according to schedule. They will also facilitate communication and collaboration amongst the other developers throughout each sprint cycle.

2.3 PROBLEM REPORTING

The problem reporting process for the train simulation system will use GitHub as the issue tracking system. Whenever a bug is identified, it will be reported as an issue in GitHub and assigned to the person responsible for resolution. The issue will be labeled with the appropriate severity level and will include a detailed description of the problem, steps to reproduce, and any other relevant information. The issue will remain open until the bug is resolved and verified by the tester.

2.4 DEPENDENCIES

2.4.1 SOFTWARE DEPENDENCIES

The system is implemented using Python programming language version 3.8 or higher and requires the PyQT5 toolkit version 5.14 or higher. Each software module communicates with each other using APIs designed in Python. The system is designed to run on the Windows 10 operating system. The following software dependencies are required:

- Python 3.8 or higher
- PyQT5 toolkit 5.14 or higher
- Windows 10 operating system

2.4.2 HARDWARE DEPENDENCIES

The system requires the use of an Arduino board to control the hardware components. The Arduino board will communicate with the Python software components. The following hardware dependencies are required:

- Arduino board

2.5 RISKS AND ASSUMPTIONS

2.5.1 RISKS

- *Risk 1:* Poor communication and collaboration among team members could lead to delays in development and poor system integration.
- *Risk 2:* Individual module testing may not accurately reflect system-level behavior, leading to potential issues in the overall system.
- *Risk 3:* Technical issues with GitHub or other tools used for collaboration and version control could cause delays or loss of work.

- *Risk 4*: Inexperience with the technology used in the project could lead to bugs or inefficient code.

2.5.2 ASSUMPTIONS

- *Assumption 1*: Each team member will be able to effectively manage their own module and communicate with other team members as needed.
- *Assumption 2*: Each team member will be able to develop effective unit tests for their own module to ensure proper functionality.
- Assumption 3: The hardware component will be able to effectively communicate with the software modules as designed.
- Assumption 4: Each team member will be able to effectively use GitHub for version control and collaboration.

3. TESTING STRATEGY

3.1 UNIT TESTING

Unit testing will be performed on each module separately. These tests are in Github in the ECE1140/train_system/tests directory. Each module's tests is located in a different file.

3.1.1 CTC

Unit tests will be used to ensure functionalities for all key functions in the module. This includes calculating authority, suggested speed, route, also would include setting track block for maintenance and altering switch positions to alter calculations for the first three variables.

3.1.2 Track Controller Software

Unit tests will be used to make sure all use cases are functional. The main functionalities to be tested are uploading the PLC file, manually changing the outputs under maintenance mode (commanded speed, traffic lights, switches, and crossings), and making sure the PLC logic is functional and provides the expected outputs under the given input (track occupancy).

3.1.3 Track Controller Hardware

Tests will be created for each key function of the controller. Tests will include successfully uploading a PLC file, getting key information from other modules, manually changing switches, lights, and railways, and sending relevant data to other modules.

3.1.4 Track Model

Unit tests will be added for every new piece of functionality or method added to the module. By the final integration, all key functionalities will be unit tested. The core pieces of functionality are the tack faults being sent and a csv being able to be successfully uploaded.

3.1.5 Train Model

Unit tests will be added for all key functionalities in the train model class (I.e., test the force, acceleration, and actual velocity calculations) as well as the failure modes (I.e., test the get() methods for engine, signal-pickup and brake failure). By the final integration, all key functionalities will be tested and communicating properly with surrounding modules.

3.1.6 Train Controller

Unit tests will be created such that all inputs and outputs are covered. Emphasis will be placed on testing vital systems, such as the PI controller.

3.2 System Testing

System testing will be performed to verify that the train simulation system meets the specified requirements. The testing will be conducted on each module individually, as well as on the system. Test cases will be designed to cover all system functionalities and scenarios. The testing will be done manually and automatically, and the results will be recorded in the test logs. Issues found during testing will be reported in GitHub and assigned to the module owner responsible for resolution. The final acceptance test will be conducted by the project stakeholders to verify that the system is ready for deployment.

4. Test Procedure

Test Case	Inputs	Expected Output	Pass/Fail	Failure Description	Tester	Date Tested
CTC						
Calculate authority for a train based off destination	Destination Station	Authority	Pass		Zach	4/7/23
Calculate suggested speed for a train	Destination Authority Other train locations	Suggested speed	Pass		Zach	4/7/23
Track Controller - Software						
Manually change outputs	Traffic lights, switch positions, crossings	Traffic lights, switch positions, crossings	Pass		Rayan	4/7/23
Wayside reads PLC file	PLC file (text file)	PLC file	Pass			
System computes traffic lights, switches, crossings based on PLC file	Track Occupancy, PLC file	Traffic Lights, switches, crossings	Pass			
Receives authority, suggested speed from CTC	Authority, Suggested speed	Authority, suggested speed on the correct block/for the correct train	Pass			
Receives failure modes and track	Failure modes	Failure modes and	Pass			

Occupancy from Track Model	(broken rail, circuit failure, power failure) and track occupancy	train presence detected on correct block				
Track Controller - Hardware						
Receive suggested speed from CTC	CTC suggested speed	Proper suggested speed	Pass		Stephen	4/7/2023
Receive authority from CTC	CTC authority	Correct authority	Pass		Stephen	4/7/2023
Wayside reads PLC file	PLC file	Correct reading and processing of PLC	Pass		Stephen	4/7/2023
PLC changes traffic color	PLC file	Changing traffic light	Pass		Stephen	4/7/2023
Receives track occupancy from track model	Track model occupancy	Correct track occupancy	Pass		Stephen	4/7/2023
Receives track status from track model	Track model track status	Correct track status	Pass		Stephen	4/7/2023
Send track status to CTC	Track status	Received from CTC	Pass		Stephen	4/7/2023
Railway crossing activate	Track occupancy	Railway crossings activate	Pass		Stephen	4/7/2023
Manually change switch position	User input	Switch change	Pass		Stephen	4/7/2023
Train Model						
Calculate force according to a commanded power and initial velocity	Commanded power, velocity, mass, grade and elevation	force	Pass		Caiti	4/7/23
Calculate acceleration according to	Calculated force, total mass	acceleration	pass		Caiti	4/7/23

calculated force value						
Calculate actual velocity of train according to change in acceleration over time	Calculated acceleration, time	Actual velocity	pass		Caiti	4/7/23
Detect failure modes of the train: engine, signal-pickup and brake failure	Engine, signal, and brake status	Engine, signal and/or brake failure	Pass		Caiti	4/7/23
Train Controller						
PI Controller outputs expected power when train velocity is lower than commanded velocity	Train speed, commanded speed	power	Pass		Tej	4/7/23
Emergency brake is triggered during train faults	Power fault, brake fault, signal fault	Emergency brake	Pass		Tej	4/7/23
Power is 0 when service or emergency brake is triggered	Emergency brake, service brake	Power	Pass		Tej	4/7/23
Train driver opens and closes doors in manual mode	Left door input, right door input	Left door, right door	Pass		Tej	4/7/23
Train driver controls lights in manual mode	Interior lights input, exterior lights input	Interior light, exterior light	Pass		Tej	4/7/23
Lights automatically turn on underground and at night	Underground, time	Interior light, exterior light	Pass		Tej	4/7/23
Train stops at destination station, announces station and opens doors	Beacon, authority	service brake, doors, station name	Pass		Tej	4/16/23
Brakes are triggered	Authority	Emergency Brake,	Pass		Tej	4/16/23

depending on authority distance		Service brake				
Track Model						
Upload Track Layout with a .csv file	.csv file containing Block Number, Block Length (m), Block Grade(%),Speed Limit (Km/Hr), Infrastructure, station side, ELEVATION (M),CUMALTIVE ELEVATION (M)	Line (blue, green, red), Switch Information, stations	Pass		Rebecca	04/07/2023
System Testing						
Dispatch train on green line	User select station, user push "Dispatch Train", Upload .csv for track layout, Upload PLC	Train moves from yard to station selected	Pass		Tej	04/06/2023
Train stops at stations along track	Beacon	Display each station on system UI's	Pass		Zach	4/6/23
Dispatch train on red line	User select station, user push "Dispatch Train", Upload .csv for track layout, Upload PLC	Train moves from yard to station selected				