Assignment 3: Multitasking with C and the BeagleBone Black: Railway Crossing Simulation

Group 20:

Ashlesha Deokar: G01374665
Aniket Anil Raut: G01387118
Mandar Chaudhari: G01393699
Ganesh Madarasu: G01413183

Github Repository: https://github.com/mandarc64/CS692_001_G20

Project Overview

This project involves modeling a railway crossing system using multi-threaded C code on a BeagleBone Black. The simulation uses push-buttons to represent sensors for an approaching train, a servo to simulate the crossing guard, two red LEDs to represent the crossing signal, and a piezo buzzer for the alarm.

System Components

- Pushbuttons: Simulate sensors for detecting the direction of the approaching train and its position relative to the crossing.
- Servo with Single Arm: This represents the crossing guard that is lowered when a train approaches and raised when it clears the crossing.
- Red LEDs: Simulate the crossing signal, blinking in an alternating pattern when a train approaches and stopping when the train clears the crossing.
- Piezo Buzzer: Acts as an alarm in the event of a possible collision scenario.

System Operations:

Train Approaching

- The LEDs start blinking in an alternating pattern.
- The crossing guard (servo) is lowered.

Train Clearing

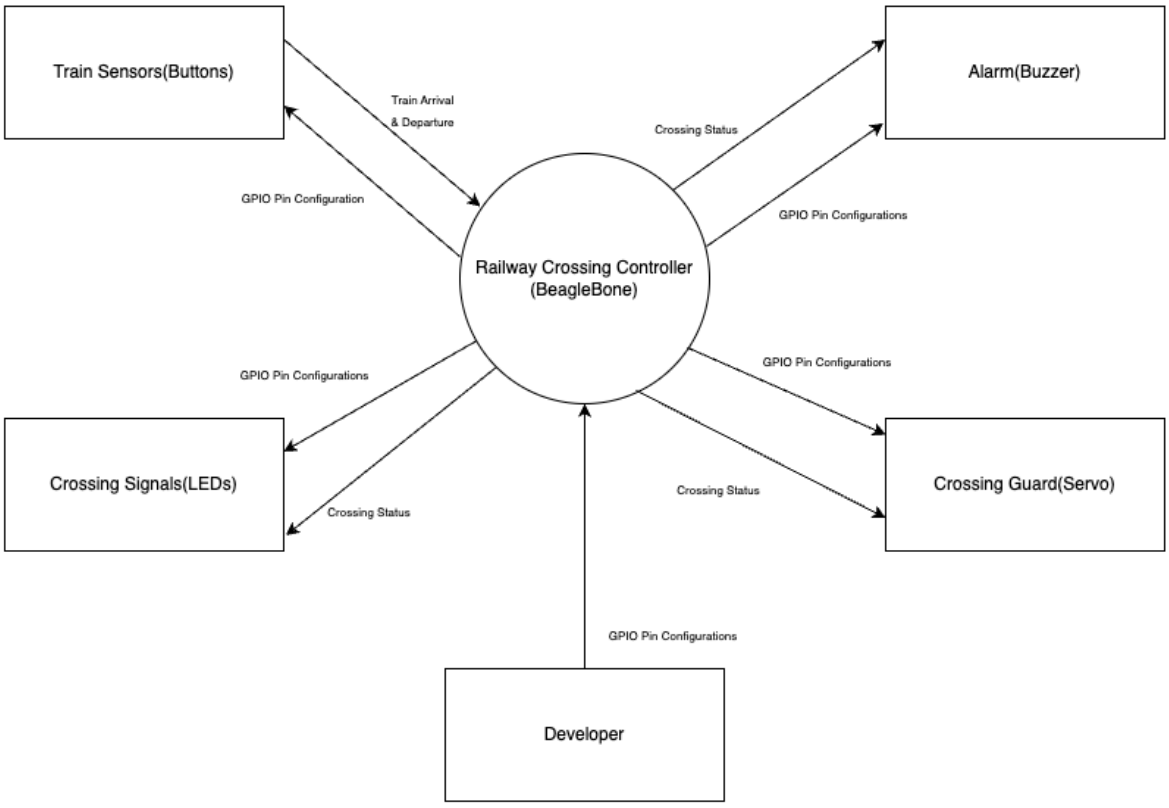
- The LEDs stop blinking 1 second after the train clears the crossing.
- The crossing guard is raised 1 second after the LEDs stop blinking.

Collision Scenario

- If the system detects trains approaching from both directions simultaneously, it ensures the crossing guard is closed.
- Both LEDs flash simultaneously in a rapid pattern.

- The alarm (piezo buzzer) is triggered.

Context Diagram:



Software Implementation:

The software is implemented in multi-threaded C code on the BeagleBone Black. It includes the following key components:

- GPIO Initialization: The GPIO pins for the LEDs, pushbuttons, servo, and buzzer are configured, and the user is prompted for the GPIO pins being used.
- Button Listener Threads: Separate threads are created for each pushbutton to detect button presses and determine the sequence of events based on the order of button presses.
- Crossing Control Thread: A thread that controls the state of the LEDs and the crossing guard based on the sequence determined by the button presses.
- Servo Control: The servo is controlled using PWM to simulate the lowering and raising of the crossing guard.
- LED Control: Functions are implemented to control the blinking pattern of the LEDs.
- Buzzer Control: Functions are implemented to control the buzzer in the event of a collision scenario.

We have also added a condition for safe operation where the system is designed to reset after 20 seconds of inactivity.

Testing and Validation:

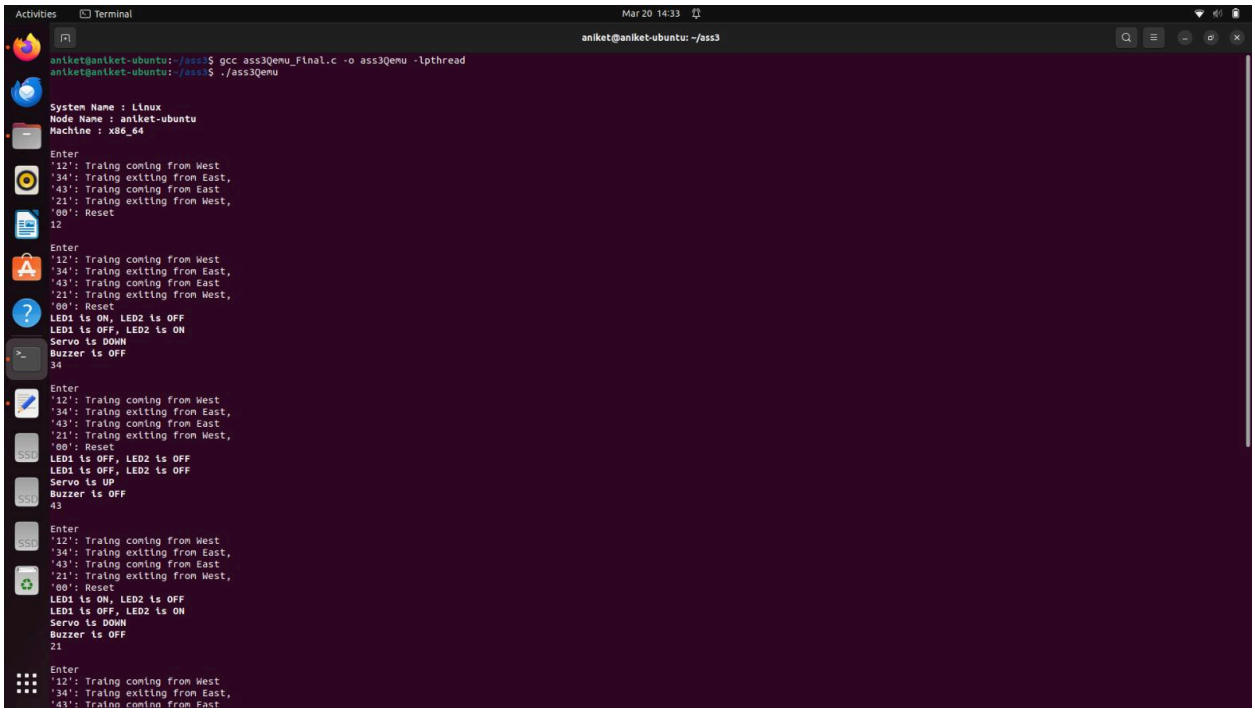
The code is first tested with an emulator to ensure correct functionality. The system's uname information is printed during initialization for verification purposes.

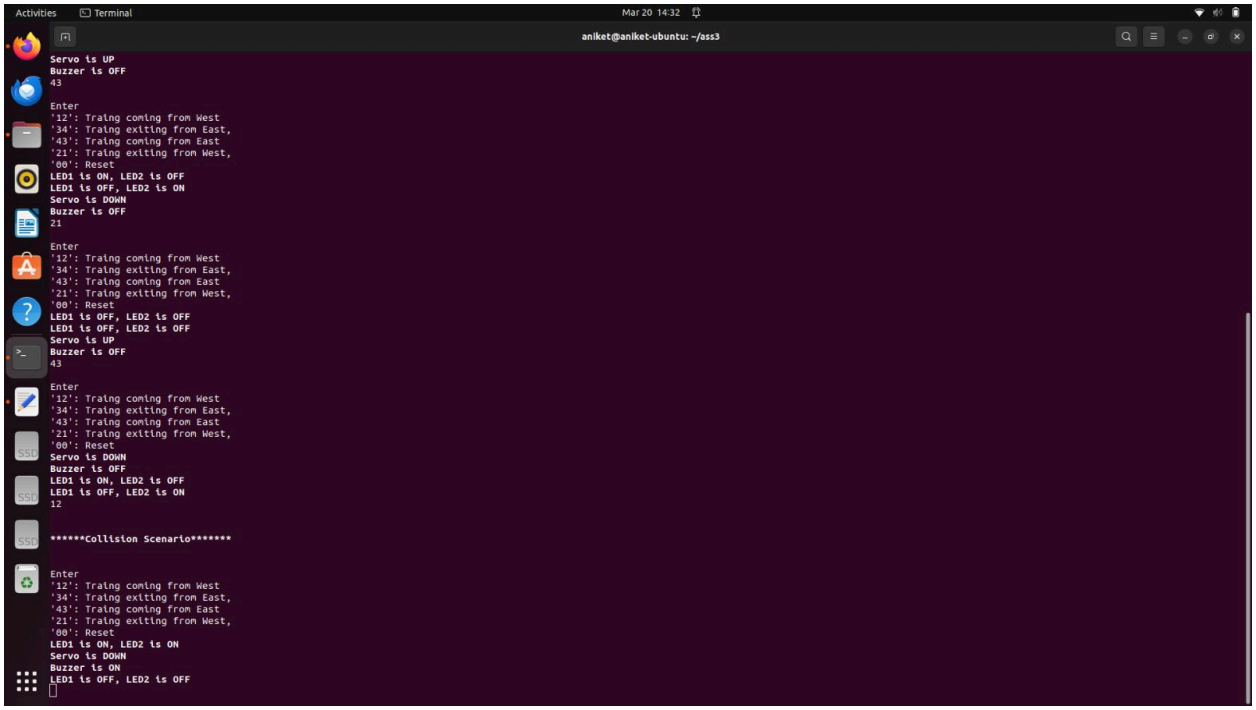
Project Outcomes:

- A functional traffic signal simulation that can be controlled via user input.
- Modular code that adheres to coding standards.
- Successful implementation and testing on both an emulator and the BeagleBone.

Supporting Materials:

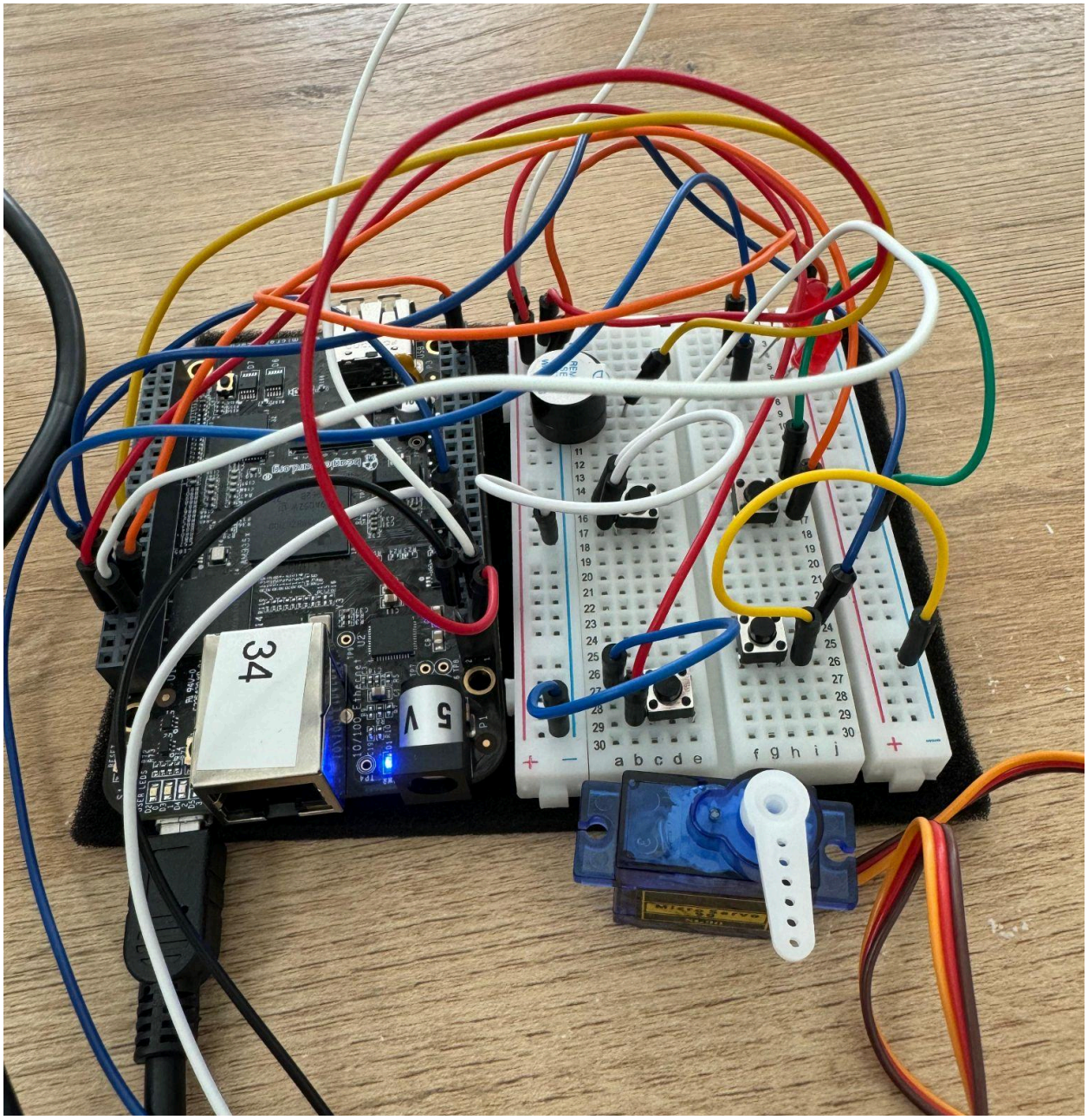
- Screenshot of Emulator Output: The screenshots show the output of the code executed on the emulator, demonstrating the simulated traffic light transitions.





- Video of Hardware Execution:

  Video Link:
  - https://drive.google.com/file/d/1EsHvuihGavH0Cx7RsbNUYYHDHDx1GRtq/view



Conclusion:

This project demonstrates the use of multi-threading in C to simulate a railway crossing system on the BeagleBone Black. The system successfully detects the direction of an approaching train, controls the crossing guard and signal LEDs, and triggers an alarm in the event of a possible collision scenario. The GPIO pins used in the simulation are configurable, allowing for flexibility in the hardware setup.