ICE A1 | u3284216

## Exploring the Problem

1)

I am tasked with creating a high-level conceptual logic system to ensure the implementation of a railway crossing is designed safely.

2)

Inputs:

* Train approaching (input from a sensor placed at a determined distance before the crossing)
* Gate lowered (input verification)
* Train departed (sensor placed a determined distance after the crossing to verify the train has passed the crossing)

Outputs:

* Lower gate (an output signal that closes the gate)
* Raise gate (an output signal that raises the gate)
* Alarm (to audibly signal that a train is approaching, and the gate is closing)
* Warning lights (lights to accompany the alarm as an extra form of warning)

3)

Not much in terms of technical constraints, given that the context of this assignment is high-level I am to assume that we have abundant resources and will be able to purchase the technology necessary to implement this logic into the real world, the technology has also been around for a long time. Economically, it will cost some money to implement, however this is trumped by the economic savings that occur by mitigating accidents as well as the social expectation of safety and rules-based roads. Environmentally, this will consume a relatively negligible number of resources. Safeguarding measures are also to be expected legally. The key stakeholders will likely be the train companies (business) train drivers and the general public. The implementation of this safety system is beneficial to all parties (maybe aside from insurance fraud artists who will now find it harder to sue over negligence by the train companies and governing body of railway safety)

## Exploring Alternatives

1)

* Logic flow 1: Simple Implementation
  + This is the most barebones interpretation, where the sensor would pick up that a train is incoming and would then lower the gates until the train has been verified as departed from the crossing
* Logic flow 2: Safeguarding
  + This implementation has a few more failsafe’s that the barebones interpretation. There will be more forms of input validation, such as verifying whether or not the gate is open, or closed, which will be determine by an operational input that stores the value of the gates current state.

My choice will be logic flow 2, while remaining simple enough to implement it prioritises safety and is more robust than logic flow 1

2)

ElectroLogIXS System by Siemens Mobility:

Uses an axle counter to detect trains, a sensor is placed at the start and end of a section of rails, the start counter increments values (per wheel) and the end sensor decrements values) if the value is non-zero then the system knows that the space is occupied, meaning the train must fully pass the sensors. To ensure safety the system uses ‘diverse redundancy’. This is where the same logic is run on two differently coded processors, they must agree on every output, otherwise the system is shut down into a safe state.

## Planning and Implementation

Sequence of tasks:

* Gates remain open
* Train is detected via sensor
* Sensor writes to a bit in data memory that stores the value (TRUE)
* When that bit reads true, it produces an array of outputs (Close gate, sound bells, LED flashes)
* The system continues as such until the departure senor detects the train and then writes to the same bit (FALSE)
* System reads the false value in the bit and send an array out outputs (Open gate, LED off, alarm off)

1. Gates Idle
   1. All outputs set to open state by default
2. Train detection
   1. The sensor detects an input
   2. Program logic writes that input into a memory bit (%M0.0) “TRUE”
3. System Reaction
   1. The PLC reads %M0.0, which will set off an array of outputs specified in an if condition (Close Gate, Bell, LED Lights)
   2. A sensor detects the gate is fully closed
      1. The program writes to a memory bit (%M0.1) to verify closure “TRUE”
      2. If closure is not verified while %M0.0 = “TRUE” program will execute outputs (notify crew, keep bells and LED’s on until manually disabled)
4. Departure detection
   1. The departure sensor detects an input
   2. Program logic writes the input into %M0.0 “FALSE”
5. Return to Idle
   1. With %M0.0 set to false the program executes the outputs according to program logic (Open gate, bells off, LED lights off)
   2. A sensor detects that the gate is fully opened
      1. The program writes to a memory bit %M0.1 “FALSE”
      2. If opening not verified while %M0.0 = “FALSE” program will execute outputs (notify crew, display “GATE FAILURE” LED sign, LED’s On)

## Logic flowchart

## Testing and Refinement

**Test Cases:**

Case 1: Normal Operation

|  |  |
| --- | --- |
| **Input** | **Expected Behaviour** |
| TrainDetected = TRUE | %M0.0 set to TRUE, all outputs activate |
| GateClosed = TRUE | %M0.1 set to TRUE, systems proceed |
| TrainDeparted = TRUE | %M0.0 set to FALSE, all outputs activate |
| GateClosed = FALSE | %M0.1 set to FALSE, systems proceed |

Case 2: Gate Closure Failure

|  |  |
| --- | --- |
| **Input** | **Expected Behaviour** |
| TrainDetected = TRUE | %M0.0 set to TRUE, all outputs activate |
| GateClosed = False | %M0.1 remains at false, emergency measures activate |

Case 3: Gate Opening Failure

|  |  |
| --- | --- |
| **Input** | **Expected Behaviour** |
| TrainDetected = TRUE | %M0.0 set to TRUE, all outputs activate |
| GateClosed = TRUE | %M0.1 set to TRUE, systems proceed |
| TrainDeparted = TRUE | %M0.0 set to FALSE, all outputs activate |
| GateClosed = TRUE | %M0.1 remains true, emergency measure activates |

Case 4: Early departure input

|  |  |  |
| --- | --- | --- |
| **Input** | **Actual Behaviour** | **Expected Behaviour** |
| TrainDetected = TRUE | %M0.0 set to TRUE, all outputs activate |  |
| GateClosed = TRUE | %M0.1 set to TRUE, systems proceed |  |
| TrainDeparted = TRUE | %M0.0 set to FALSE, all outputs activate | Input should be ignored until certain |
| GateClosed = FALSE | %M0.1 set to FALSE, systems proceed |  |

There are a few issues with the system. My idea for sensors were pulsing laser (or IR) sensors that upon detection will continue to emit a beam. These would detect the instance that the front of the train activates the sensor up until the train clears it. One main issue for is that longer trains may be able to activate both sensors at the same time. Additionally, there is no real failsafe for is one of the train detection sensors gives false input. It also would not be able to distinguish between a train and say some random object placed in front of it. An Idea I had to mitigate the false sensor (false departure reading) would be to have a timer attached to the first detection sensor so that It knows how long the train would take to pass the sensor at its current speed, so that when the output sensor is triggered it has to wait minimum that amount of time before it can send the output commands to raise the gate, Though this does not entirely mitigate the problem. The axle counter from the ElectroLogIXS system are a much more reliable way of detecting the train and verifying that it has actually departed before opening.