

SWT PWR Project Report

SWT-PR-M-Group-C

SWT-PR-M

Winter Semester 2020/21

Group C

Deepika Arneja

Student Number: 1987471

Degree Course/Semester: ISoSySc/4

Mahender reddy Sheri

Student Number: 1985741

Degree Course/Semester: ISoSySc/4

Supervisor: Yugene Yepp

Version: 15.02.2021

Abstract

The existing TS2 signal library only supports UK and France railway signals and signals in these countries are displayed on left side of the track layout

The significance of this project is to create German signal library and extend the support in TS2 editor.

As part of this project, I acknowledged to various types of German railway signals each of which is used in different pattern for different purpose. German signals are displayed on the right side of the track plan in travel direction.

A German signal library with Signal Aspects and Signal Types is defined and signal states as implemented as per respective signals functionality.

The current project deals with train game simulation, extension of the already available features i.e, including German signal library and implementation of the German signals in the simulation. Realistic train engine movement which includes energy conscious behavior, controlling the rushed behaviour of train by altering the current speed profile, including the weights and directionality for different rolling stock types. Design overview of the automated traffic controller in the simulation with simple flowcharts and abstract methods.

Contents

Abstract	1
1 Problem Description	5
1.1 Context and Goal	5
1.2 Methodology	5
2 Problem Analysis	7
2.1 Related Work	7
2.2 Project Organisation	9
3 Design	11
4 Implementation	17
5 Evaluation	21
6 Conclusions	25
References	27
A Sprint Goals	29
Ehrenwörtliche Erklärung	31

1 Problem Description

1.1 Context and Goal

Goal of this project is to create German Signal library and include it in TS2 editor. User should be able to create track plan in TS2 editor using German signal library and run simulation.

German signals should be displayed on right side of the track platform whereas signals of UK and France should be displayed on left side of travel direction. Any track plan created with German signal library should load in TS2 editor and signals should work as defined while simulation is running.

The project aims at train game simulation with realistic train engine movement by changing the current speed profile of the train considering the energy conscious behavior of train and also limiting the directionality of the trains to only certain types. The additional parameters for the rolling stock train types such as weight, engine power are added in order to take advantage of those parameters for acceleration calculation. Design overview of the Automated traffic controller is made for further implementation, scoring factor for penalty with certain constraints such as shortest route possible, distance covered, energy conscious behavior, energy consumed, braking distance are considered into the picture.

These changes helps in energy saving behavior of the train with real world constraints as well as it gives a well understanding of the movement of the train throughout the journey. It helps to restrict the movement of trains to set services by limiting the directionality of the trains.

We also aim at providing possible suggestions for updates such as including emergency button, reporting the energy consumption at regular intervals, considerable extensions in display such as acceleration along with speed etc. for possible recommended extensions.

Overall project aim can be achieved to a maximum extent if we are not able to reverse trains such as cargo, if factors such as weight and engine power are able to successfully added to the traintypes and are actually reflected in the actual simulation while calculating the acceleration of the train.

1.2 Methodology

The above stated problem of German Signal is achieved by studying types of German Signals, their design patterns, and their functionality .

Total five different signals are defined. Signal aspects are created same as produced by Viessmann [11]. Each signal displaying different light and pattern[13]. A track plan is created and all types of signals are placed as per their definition

Types of Signals -

- ¹ Block Signal - A block signal is used at the entry or exit of a normal block.
- ² Departure Signal - A departure signal is place at the exit of a platform to inform driver whether they can depart from the platform or not.
- ³ Entry Signal - An Entry signal is placed before the entry of a platform (block) to inform driver whether they can enter the platform or not.
- ⁴ Distant Signal - A distant signal is placed before the main signal to inform driver about the state of main signal.
- ⁵ Shunt Signal - A Shunt signal is placed to separate main segment from yard segment.

The above stated problem mainly can approached through a deep study of the speed profile of the existing train and various research works related to the movement of the train as well as the

acceleration and deceleration strategies considered at different contexts with various parameters through trial and error basis of the acceleration and deceleration equations.

The directionality of the trains can be managed by studying the reverse function of the trains and various limitations for the reverse errors of the traintypes. Effective equations for braking distance and braking force are calculated in order to represent them to the user in message logger for change in driving behavior.

Possible abstract methods commented along with code also stating the possible references for the further developers for complete implementation.

2 Problem Analysis

2.1 Related Work

Railway Signalling is an important element. Existing implementation only supports UK and France signal library. There was no provision to create track plans with German signals. Also, existing Signals are displayed on left side of the track plan.

As part of this project, German Signal library is created and included in TS2 editor. German Signal library uses Hauptsignal/Vorsignal [12][1]. Block, Departure, Entry, Distant and Shunt types of signals are defined. Different types of signals can be added in the track plan depends upon their definition and they work based on the Signal State and conditions defined in Signal library. German Signals are placed on right side of the track and also supports functionality of reverse trains.

Due to TS2 editor constraint SHUNT signal is not implemented. Shunt signal pattern is not included in TS2 editor and shunting includes manual efforts in real-time scenarios. Only SHUNT Signal Aspect is added in library.

Current prototype is a train game simulation with python at client side and go language at server side, it is organised with certain features i.e, setting services, routes, trains, traintypes, penalty, simulation between stations. A demo file with proper services, routes, simple track plan is provided for the demo simulation and working of the project. However context regarding the bahnc compiler is also provided in the existing project but key information related to the trains, traintypes, tracks, tracklayout etc., are studied from bahndsl thesis [1]. Actual working of the current simulation is based on normal parameters with rushed behavior of train.

A realistic train engine movement is studied based on the requirements provided i.e, improving the trains acceleration, speed profile, and including weight, engine power as well as change in directionality of traintypes. Acceleration profile currently depends on the standard acceleration value provided in the editor without considering the factors such as engine power, weight which are studied based on the engine to power ratio of a vehicle at a constant power [2] though the train not use constant power at all times accelerated values are chosen based on the current speed.

There are many factors considered for the performance enhancement of the train while moving, where rolling resistance, weight, acceleration resistance, tractive resistance with an analytical approach by Chul et al.[3] for electric run train systems provides an insight of using various factors for improving performance out of which at present we have considered weight, engine power. Although the trains provide 20 times more energy saving than other locomotives [4] due to less resistance at variable speeds, low friction between the wheels and the tracks where it may also be changed across turnings, tunnels, cuttings, bridges we may consider some more factors for improving energy efficiency. It also states that at constant speed it only consumes 5% of the actual energy in steel wheel vehicles which points out that improving the timing of constant speed also helps in energy saving behavior of train.

Basic information of the details of equations are learned [5] with including factors such as speed, time, weight etc., where acceleration, deceleration, braking distance with speed is the interdependent factor for all those equations. It helped in understanding the importance of speed in energy saving behavior of vehicle.

Distance plays a key role in safety and energy efficient systems [6], where in the train movement distance to the next signal, next speed limit, next station are to be considered precisely for the proper deceleration of the train to keep the train in limit. It also states that the distance kept by the current train from the preceding train while moving is necessary for stopping at a fixed point as well as regulating the speed. This work helps in the study of distance signal for the standard braking system of the train.

Energy efficient driving study of other vehicles says that mass, braking, anticipating traffic or delays, weather, fuel type, energy losses [7] considered for improving fuel efficiency which is also a considerable factor for vehicles other than electric run where for e-vehicles regenerative braking is considered for saving energy while braking [8].

Braking plays a major role in controlling the train where the train is actually accelerated based on the standard braking value as the speed of the train must not surpass beyond the limit of braking which may cause a brake failure. For e-vehicles quick brakes may also help in saving energy using regenerative methods where the current train in the simulation brakes as slow as possible and so changing the braking behavior is an aspect to be considered for the implementation.

All the above factors states us the relative dependency of each parameter such as weight, speed, braking, distance for energy saving behavior of the train making the importance of study of such factors on realistic train engine movement.

Directionality of the train is a limitation of the existing project where all types of trains are allowed to reverse but there are certain factors to be considered for such as cargo trains cannot be reversed. Weight of the train is considerable factor for movement as previously stated as for light weight vehicles it effects a lot with the centre of mass of the weight in cargo, passengers in vehicles [9] and so respectively it effects to a minimum extent for heavy weight vehicles.

Automated traffic controller is the system for automating the trains in shortest routes possible by considering the deadlocks, interlocking as the reliable factors as well as existing penalty system is given with only certain factors such as train delays, arrival in wrong platforms where energy conscious behavior is a limitation to it. These studies from the existing project made clear to design a new strategy for scoring system as well as considerable additional factors for automating such as emergency braking which is twice as effective as normal braking.

Apart from the theoretical background knowledge on all these factors for realistic train engine movement technical aspects are considered based on the existing project i.e, a proper study of client side and server codes as well the example demo files in detail. Python code at the client side lacks in full implementation of context menu widgets in train.py where reset train service, split train service are partially implemented. At server side the actual functioning of the train is designed i.e, how the train should move and where the train should stop. But as energy saving factors are not considered the simulation consumes more energy than what is needed.

The speed function in the trains file has to be fixed with as many trial and error cases as possible as there are many min, max functions in depth which alter from time to time, position to position, in the simulation. Unfortunately parameters, statements in the speed function for setting the limitation of the user inputs is a backlog without effective control on whether the type of data used in the calculation is actually reasonable or not.

Hopefully the existing system supports the addition of new parameters in traintypes but it needs a special function to be defined for trains (eg. Acceleration). The message logger at the server side is also limited with few test cases which does not support the inclusion of new simulation messages in the server side for the user display.

Editor of the existing project helps us to create new trackplans, routes, services, trains for the simulation but the editor needed to be improved for realistic trackplan design i.e, evaluating the trackitems while updating the scenario which indeed a time consuming aspect right now, improving the editor helps in concentrating on developing the code with as many simulation trials as possible with various track plans.

2.2 Project Organisation

Project organised based on the work packages such as research, implementation, debugging, testing, documentation.

Hours	Begin	End	Work
32	16.11.2020	30.11.2020	Project and Railway signal domain research
28	01.12.2020	14.12.2020	understanding German signals functionality
28	15.01.2020	11.01.2021	Implementing German Signal library
42	12.01.2021	27.01.2021	Simulation and Shunt Signal documentation
60	26.02.2021	14.02.2021	Testing and Documentation
Total Hours:		190	

Table 1: Work

Hours	Begin	End	Work
25	16.11.2020	12.12.2020	Research on project and Reverse implementation
30	12.12.2020	06.01.2021	Studying Energy Conscious, weights implementation
20	06.01.2021	21.01.2021	Debugging code and Speed profile study
25	21.01.2021	08.02.2021	Energy Conscious implementation
30	08.02.2021	14.02.2021	Testing and Documentation
Total Hours:		130	

Table 2: Work

Date	Achievement
12.12.2020	Directionality implementation
26.12.2020	Design of energy Conscious
15.01.2021	Weights and Engine power Implementation
22.01.2021	Acceleration calculation
01.02.2021	Deceleration calculation
12.02.2021	Automated traffic controller design
15.02.2021	Project Submission

Table 3: Milestones

3 Design

German signal library supports below signal types -

GERMAN_BLOCK_MOVE, GERMAN_BLOCK_STOP, GERMAN_DEP_LOW, GERMAN_DEP_MOVE, GERMAN_DEP_STOP, GERMAN_ENT_LOW, GERMAN_ENT_MOVE, GERMAN_ENT_STOP, GERMAN_DIS_LOW, GERMAN_DIS_MOVE, GERMAN_DIS_STOP, GERMAN_SHUNT_STOP and GERMAN_SHUNT_MOVE.












Signal Type	Description	Display Pattern
GERMAN_BLOCK_MOVE	Informs driver to move into the next block	
GERMAN_BLOCK_STOP	informs the driver not to enter the next block and stop before this signal	
GERMAN_DEP_LOW	informs driver to slow the speed and prepare to stop before the next signal.	
GERMAN_DEP_MOVE	informs driver to move to exit the platform and move into the next block.	
GERMAN_DEP_STOP	informs driver not to exit the platform and stop before this signal	
GERMAN_ENT_LOW	informs driver to slow the speed and prepare to stop before the next signal	
GERMAN_ENT_MOVE	informs driver to move into the platform	
GERMAN_ENT_STOP	informs driver not to enter the platform and stop before this signal	
GERMAN_DIS_LOW	informs driver to slow the speed and expect stop at main signal	
GERMAN_DIS_MOVE	informs driver to move into the subsequent main block with full speed	
GERMAN_DIS_STOP	informs the driver not to enter the platform and stop before this signal	

Figure 1: Signal Types with description and display pattern

Figure 1 describes the signal types implemented in TS2 editor with display pattern. German Signal libraries are designed to display on right side of the track and it also supports reverse functionality. Signals with reverse set to true are placed on the left side of the track and are applicable to trains coming from right side of track.

Signals are placed on a track based on their functionality and respective signals display pattern and unique light signals gives clear indication to driver what actions to be taken further.

Figure 2 describes the SHUNT Signal design approach. Shunt signals are used to inform driver if shunting is allowed or not at the moment. Shunting is used to add wagon, locomotives, change platform etc. A train has to move into yard line from main line in order to perform shunting operations. A LOS (Line of Shunting) board is added to a point to define the shunting limit, which is missing in TS2 editor tool. Shunt Signals are placed on each line to inform driver to ALLOW shunting if White and STOP shunting if Red. Shunting operation also involves lot of manual efforts in real-time.

Figure 3 Represents the general flow of realistic train engine movement where it is divided into directionality and energy conscious. While directionality deals with the reverse functionality of rolling stock types energy conscious deals with adding of parameters such as weight and engine power to the traintypes as well as change in speed profile. Using the weight and engine power we derive certain equations for the acceleration and deceleration values.

For limiting the directionality of the train the reverse functionality is considered where the cargo trains are limited to only one direction. Here reverse function deals with changing of the train

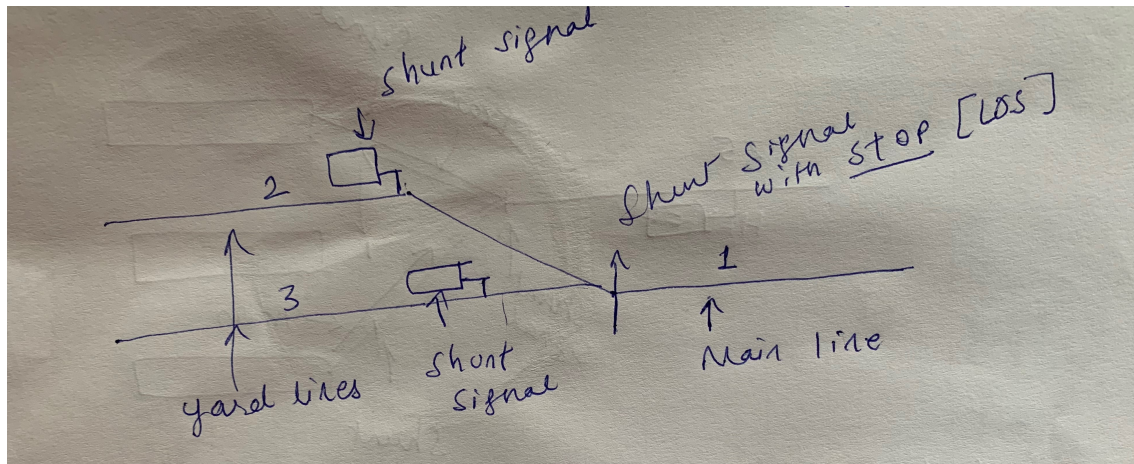


Figure 2: Shunt Signal approach

direction by setting the current head of the train as its tail and vice versa but in order to implement the function it should satisfy the constraints such as the train type code must not be cargo or other similar traintype codes which are not suitable for reverse and secondly the train must not be in a running position i.e, the train must reduce the speed to zero.

Initially design was considered to include the weight of the train by calculating it using wagon type and wagon count but later stages it is combinedly provided with other traintype parameters such as stdAccel, maxspeed by also adding engine power.

The primary responsibility of the design is to consider the factors of weight and engine power for the calculation of acceleration and also set the directionality of the train. All these factors finally deals with energy saving behavior of the train.

The design must be successfully able to import and export the additional parameters of the train-types without disturbing the existing parameters. The train speed profile must be changed by maintaining constant speed for long intervals.

The design is limited to the user inputs where the weight, engine power are expected to be realistic from the user it does not provide any further conditional statements or warning systems to the user in order to update the unrealistic data.

For acceleration power to weight ratio is considered along with other equations as the acceleration may not be same throughout the simulation and so the current acceleration is designed in such a way that the equation value changes depending on the speed at different intervals which helps in balancing the speed .

The braking design is considered with for making the train to stop based on the target distance and the current speed such that the target distance is set to minimum possible which makes the train to decelerate quickly. There is also a braking information provided in the design.

The energy consumption is an additional factor added to the context menu widget but not yet fully implemented as it is a abstract method right now without considering any of the factors such as speed, engine power, time travelled, fuel type.

The current design focuses on maintaining constant acceleration for long time, low change in acceleration and high change in deceleration i.e, applying brakes as quickly as possible which finally results in pickup speed needed for acceleration for approaching the destination, hence it is a limitation of the current project which is needed to be simplified with change in non-linear behavior.

These factors resulting in changes around acceleration in fact consumes a little more energy during initial acceleration however it overcomes it with increasing the timing of constant acceleration, The

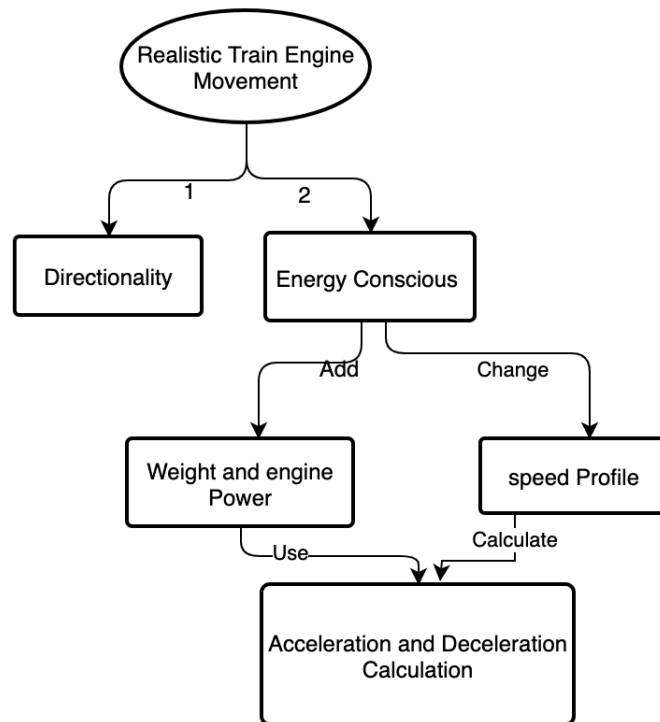


Figure 3: Flow of Realistic Train Engine Movement

acceleration is best suitable for electric run trains as these deceleration makes the train to store the energy while braking by using regenerative braking, Whereas for fuel run vehicles it does not take advantage of that and thus incurring in loss of certain amount of energy. However the current design helps out the use to extend the features towards energy conscious more precisely and forms a novel design for improving the performance for further developers.

Current concentrates on change in acceleration values as minimum as possible leaving behind certain factors such as initial hike in acceleration after a sudden braking which is relatively higher than the existing project. The design showcases all kinds of abnormalities that can be occurred by changing the acceleration and deceleration values which considers additional factors for change in speed profile which in fact hints the developers using novel ways of improving the train performance.

```

1 #possible value restrictions to the user for weight and engine
  power for realistic values
2 if t.TrainType().Weight < 700 && t.TrainType().Enginepower <
  5000 {
3     weight := 700
4     enginepower := 5000
5     fmt.Printf("Engine power and weight are set to
      default values %f, %f due to inappropriate inputs ",
      enginepower, weight)
6 } else if t.TrainType().Enginepower < 5000 {
7     enginepower := 5000
8     fmt.Printf("Engine power is set to default value %f due
      to inappropriate inputs", enginepower)
9 } else if t.TrainType().Weight < 700 {
10    weight := 700
11    fmt.Printf("Weight is set to default value %f due to
      inappropriate inputs", weight)

```

```

12     } else {
13         weight := t.TrainType().Weight
14         enginepower := t.TrainType().Enginepower
15         fmt.Printf("weight is %f and engine power is %f",
16             weight, enginepower)
17     }
18     acceleration := math.Max(-t.TrainType().EmergBraking,
19         math.Min(1/secs*(targetSpeed-t.Speed), math.Max(t.
20             TrainType().StdAccel, math.Abs(enginepower)/(t.Speed*
21                 math.Abs(weight)))))

```

Automated traffic controller design is proposed considering the given criteria where it is divided into Setting the route automatically i.e, shortest route possible, designing scoring pattern, from the realistic train engine movement considering the emergency brake where applying emergency brake by user should automatically effect the movement of other trains in the route and provide possible warning message for the user.

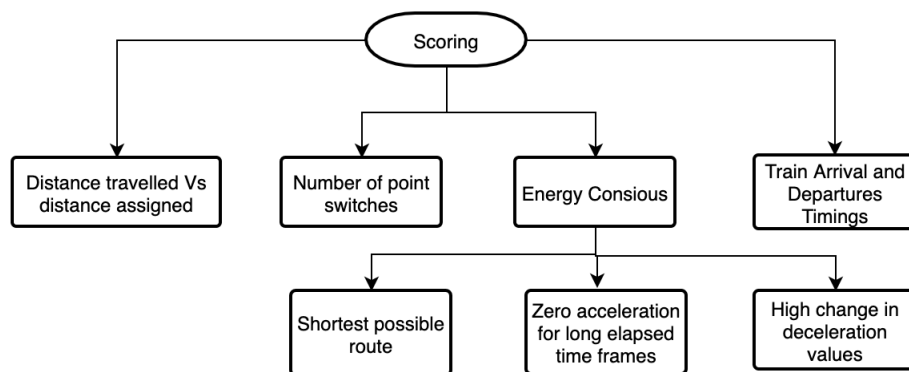


Figure 4: Scoring design

Below are few abstract methods for route implementation and Atc.

```

1 Route_grantability()
2   if route i is grantable to train
3       grant route
4   else assign another grantable route
5 Note: for granting routes the route must be free of
       interlocking conflicts and dead locks as well as possible
       shortest route at that particular time frame.

1 Interlocking()
2   assign a[i] route
3   lock the route a[i] till next station
4   if( services > 1)
5       if(service of a[i] does not conflict with service of a[n])
6           proceed_service
7   else
8       assign prior service a[i] in route i and hold service n in
           possible nearest signal.
9   Clear_route().

```

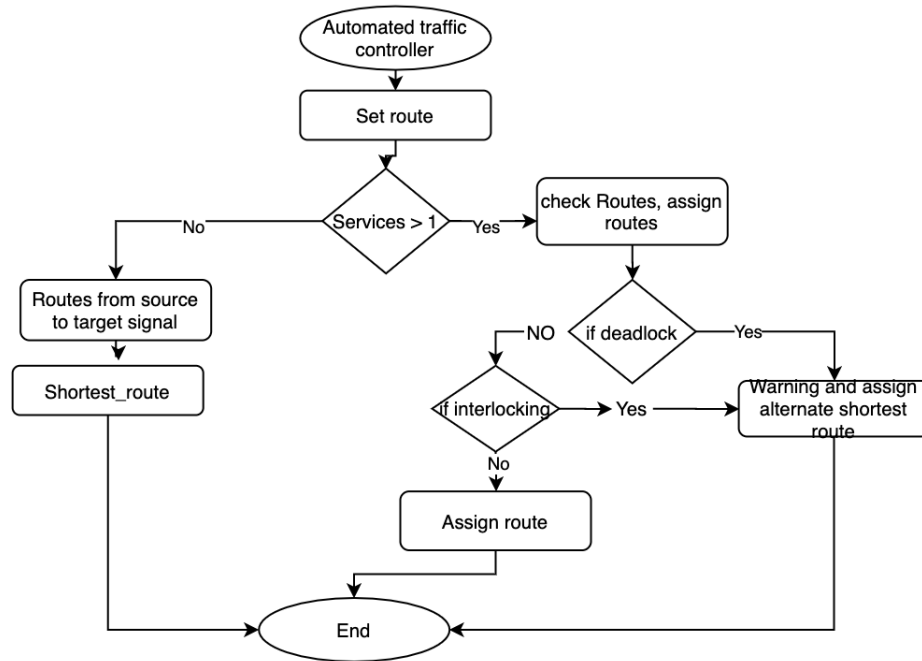



Figure 5: Set Route ATC

Interlocking is the key criteria of the train where interlocking conflicts result in collision of trains and so the route is locked for a particular train as soon as it is assigned.

```

1 Set_route():
2     Source_signal() to target_signal()
3     find_routes from Source_signal to target_signal
4     if (route_count > 2)
5         assign route
6     else
7         length(line items in route a[i] to a[n])
8         compare length(a[i] to a[n]) quick sort
9         get shortest of a[i]
10        assign a[i]

```

Possible routes are set based on the given source signal and target signal with minimum possible shortest distance.

```

1 route_clear():
2     if signals in service[i] in route [i] coincide with signals
        in service[j]
3         return false
4     elif source signal state != "stop"
5         return false
6     elif
7         for blockid in blockids
8             if blockid occupied(block_id):
9                 return false
10            end
11        end
12    else
13        return true

```

Route_clear is helpful in determining whether the route is clear to proceed or not and hence it reflects in setting of new routes to the trains provided by the user.

Energy Consumption is the key feature to be studied in modern day transportation for finding optimal ways of reducing the energy consumption of trains. Kevin in his thesis work[14] has described the Continuous-time problem is the the issue for the train movement and explained about the force acceleration needed for a train to move

i.e, $F = \text{mass} * \text{acceleration}$ but in force there are other factors involved i.e, FM: the locomotive's traction force at the wheels Frr: the rolling resistance FL: the aerodynamic resistance Frs: the gradient resistance

and the actual FM adds up all the other forces we may include these factors in the calculation of force on locomotive and theoretically total energy consumption is the sum of all forces F-total for a distance from point x1 to x2. i.e, mathematically integration of F-total from x1 to x2. Other than this there is other way proposed for the calculation energy consumption which based on time and power where power equals to speed times F-total and the energy consumption is integration of power from time zero to time t.

where the units are as follows energy consumption in joules, power in watts, speed in m per sec and force in newtons. And so for the scoring of the system estimated energy consumption is calculated based on the factors we have with an estimated speed and in the simulation for the actual speed and time values the same are calculated if there is a positive change in E value then penalty may be raised and if there is a negative change it indicates less energy consumption in deed adding score value for saving energy.

4 Implementation

The proposed design is implemented by a detailed study of the existing code where for each component to be implemented considerable factors in the front end and back end is noted along with the required changes.

In order to implement the signals defined during designing, a signal library is created with "Signal Aspects" and "Signal Types".

A Signal Aspect defined the display pattern of the signal like lamp colour, lamp shape, line style and action to be taken by driver. Signals give clear instructions to driver regarding further actions to be performed.

Signal Aspect Definition-

```

1 "GERMAN_ENT_MOVE": {
2     "__type__": "SignalAspect",
3     "actions": [[0,999]],
4     "blink": [],
5     "lineStyle": 0,
6     "outerColors": [],
7     "outerShapes": [],
8     "shapes": [1, 1, 0, 0, 0, 1],
9     "shapesColors": [
10         "#808080", "#808080",
11         "#000000", "#000000",
12         "#000000", "#00FF00"]
13 }
```

A Signal Type defines a signal that can display several aspects depending on condition. Signal State is defined for each signal type and it is combination of signal aspect and conditions to have this signal displayed.

Signal Type Definition-

```

1 "GERMAN_ENTRY": {
2     "__type__": "SignalType",
3     "states": [{
4         "__type__": "SignalState",
5         "aspectName": "GERMAN_ENT_MOVE",
6         "conditions": {
7             "NEXT_ROUTE_ACTIVE": [],
8             "NEXT_SIGNAL_ASPECTS": [
9                 "GERMAN_ENT_MOVE",
10                "GERMAN_BLOCK_MOVE",
11                "GERMAN_ENT_LOW",
12                "GERMAN_DEP_MOVE",
13                "GERMAN_DEP_LOW",
14                "GERMAN_DIS_STOP",
15                "GERMAN_DIS_MOVE",
16                "GERMAN_DIS_LOW" ] ,
17            "TRAIN_NOT_PRESENT_ON_NEXT_ROUTE": []
18        }
19    }
20 }
```

GERMAN_ENT_MOVE signal will be displayed on signal "5" when "Next route is active", "there are no trains anywhere" and "Next signal shows MOVE, LOW or Distant Stop".

Signal Definition

```
1 "5": {
2     "__type__": "SignalItem",
3     "conflictTiId": null,
4     "customProperties": {
5         "ROUTES_SET": {
6             "GERMAN_ENT_STOP": [
7                 "2"
8             ],
9             "TRAIN_NOT_PRESENT_ON_ITEMS": {
10                GERMAN_ENT_STOP": [
11                    "4",
12                    "3"
13                ],
14                "TRAIN_PRESENT_ON_ITEMS": {}
15            },
16            "maxSpeed": 0.0,
17            "name": "32",
18            "nextTiId": "6",
19            "previousTiId": "4",
20            "reverse": false,
21            "signalType": "GERMAN_ENTRY",
22            "tiId": "5",
23            "x": 190.0,
24            "xn": 150.0,
25            "y": 0.0,
26            "yn": 5.0
27 }
```

To change the signal side to be displayed on right side of the travelling direction of track plan, coordinates are updated . Also, conditions are defined to display signal on right if signal is "German" otherwise on left side .

```
1 if self.name.startswith("GERMAN"):
2     p.drawLine(2, 0, 2, 7)
3     p.drawLine(2, 7, 8, 7)
4     r = QtCore.QRectF((i // 2) * 8 + 8, -((i % 2) * 8 - 11),
5         8, 8)
6     brush.setColor(QtGui.QColor(self.outerColors[i]))
7     p.setBrush(brush)
8     self.drawShape(p, self.outerShapes[i], r)
9 else:
10    p.drawLine(2, 0, 2, -7)
11    p.drawLine(2, -7, 8, -7)
12    r = QtCore.QRectF((i // 2) * 8 + 8, -(i % 2) * 8 - 11, 8,
13        8)
14    brush.setColor(QtGui.QColor(self.outerColors[i]))
15    p.setBrush(brush)
16    self.drawShape(p, self.outerShapes[i], r)
```

The parameters weight and engine power are added to the editor with supportable methods at the client side for importing and exporting them. A method for weight and engine power along with property and the setter function is defined as below.

```

1 def enginepower(self):
2     """
3         :return: enginepower of the train in horsepower
4         :rtype: float
5     """
6     return self._enginepower
7
8 @enginepower.setter
9 def enginepower(self, value):
10     """Setter function for the enginepower property"""
11     if self.simulation.context == utils.Context.
12         EDITOR_TRAINTYPES:
13         self._enginepower = value

```

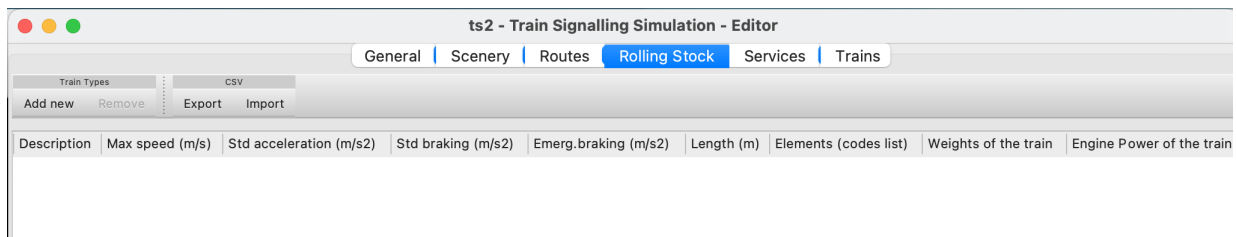


Figure 6: weight and engine power

Partial implementation of the emergency brake is also done with front end as shown below.

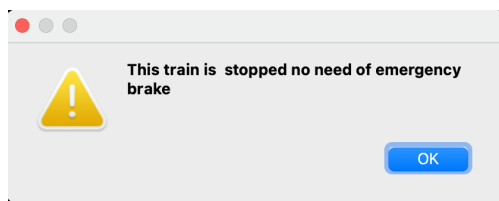


Figure 7: Emergency break

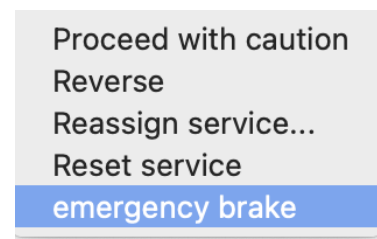


Figure 8: Emergency break option

However the implementation of emergency brake button is not fully implemented where it must be synchronized along with the automated traffic controller along with the actual functionality of decelerating at the provided value.

The reverse functionality is implemented by setting the conditional statements to the existing code as such that the actual reverse functionality does not proceed without verifying the conditions mentioned. The traintypes with codes cargo are restricted to take reverse and also warned with a message in message logger for the user.

Acceleration and deceleration values are calculated and finalised based on the trial and error methods for min max functions provided in the snippets. Acceleration equation is considered based on the related work provided with references.

```
1      acceleration := math.Max(-t.TrainType().EmergBraking ,
2      math.Min(1/secs*(targetSpeed-t.Speed), math.Max(t.
          TrainType().StdAccel,t.TrainType().Enginepower/(t.
          Speed*t.TrainType().Weight))))
```

Here the above context is implemented on several breakdowns of the equations by swapping the values of acceleration in the equation provided. Though the mentioned engine power is constant and regarded same throughout the calculation the actual values are taken based on the speed of the simulating train.

All the inner values set to min moves the train at constant acceleration throughout the simulation. And all the values set to maximum simulate the train with as high speed as possible without considering the speed limit factors. All the math values set to minimum makes the train to stop. These are few observations noted down during the implementation which are useful for change in speed values accordingly.

The deceleration is based on the standard braking and distant signal distance i.e, distance to the next signal i.e, which is used further in minimizing the speed of train.

```
1      distanceSignal = (math.Pow(t.Speed - t.TrainType().
          StandarddBraking*secs, 2)-math.Pow(t.ApplicableAction().
          Speed, 2))/(2*t.TrainType().StandarddBraking) + (t.Speed
          * secs / 2)
```

Above formula is defined in the existing project but in order to apply brakes as quick as possible we refer to the standard deceleration formula [10] by thorough study of various research works. Below is the equation

```
1      t0 = 2/k
2      stddec = math.Abs((t.Speed*t.Speed)/2*(targetDistance - t.
          Speed*t0))
3      d := math.Min(0.5*t.TrainType().StdBraking*math.Pow(secs,
          2), 0.5*math.Pow(secs, 2)*stddec)
4 Note: However the above d value for the train estimates when to
      stop the train and that particular equation best suites for
      quick deceleration but it accelerates at same speed as well i
      .e, quickly
```

K is the correction coefficient of the trainlength, here it is 1. The values from the above equation helps the train to stop the train as quick as possible.

While implementing the weight parameter we have encountered attribute not defined error which made to define the parameters at many places including the editor and also to set the default values for them. The error was corrected by correctly defining the parameters at server side i.e, capitalize the initial letter in the data structure definition of traintypes.

Directionality restrictions of the train is maintained by conditional statements in the reverse function.

Performance of the system as per the energy conscious is achieved but it can be improved further by adding additional factors and the values defined by the user are also assumed to be realistic moreover the braking distance is calculated and displayed to the user but it is restricted by the message logger for the display as there is need of changing the testcases further.

5 Evaluation

The implemented German Signal library is evaluated by creating new track plan in TS2 editor. Once the German Signal Library is placed in ts2-client/data folder an, reloading the library in TS2 editor should reflect German Signals as an option to choose from.

Properties	
Property	Value
Type	SignalItem
id	5
Name	32
Position	(190, 0)
Maximum speed (m/s)	0
Conflict item ID	
Reverse	false
Signal Type	GERMAN_ENTRY
Berth Origin	FR_BAL_S_A_...CLI_RCLI_VL
No train params	FR_BAL_S_A_R_RCLI_VL
Train Present Params	FR_BAL_S_A_VL_TP
Route set params	GERMAN_BLOCK
	GERMAN_BLOCK_TP
	GERMAN_DEPARTURE
	GERMAN_DISTANT
	GERMAN_ENTRY
	UK_3_ASPECTS
	UK_3_ASPECTS_ACFR

Figure 9: Signal Type in TS2 editor

To test the features, choose German Signals from Signal Type and define other parameters in scenery. Set routes, add train and services. Validate, Save and reopen the created track plan in TS2 to test the display and functionality of signals.

Signals should be round in shape and should be displayed on right side of the track plan. Display pattern of each signal should match as defined in Figure 1.

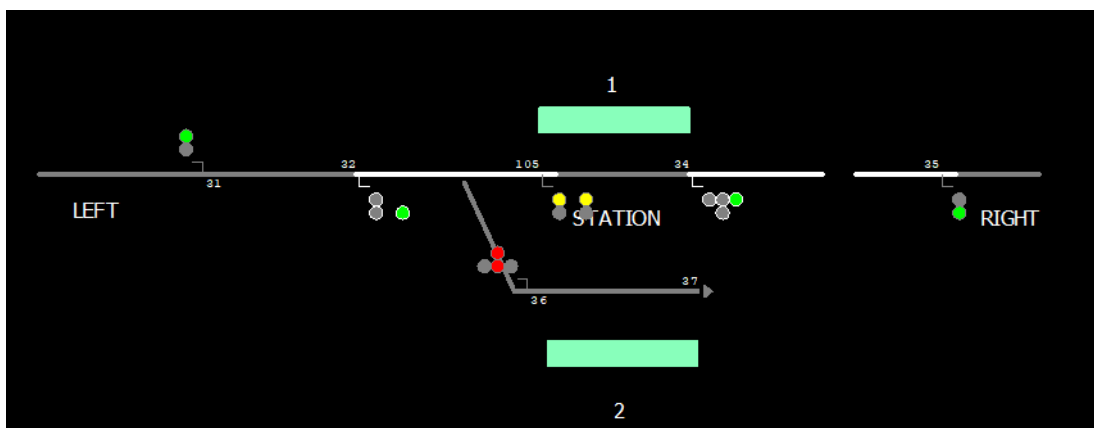


Figure 10: Signal Type in TS2 editor

Activate the desired route and run simulation. Signals should updated to Move, Stop or Low speed based on the Signals State and conditions defined with track plans. Signals should change their colour as defined in figure 1.

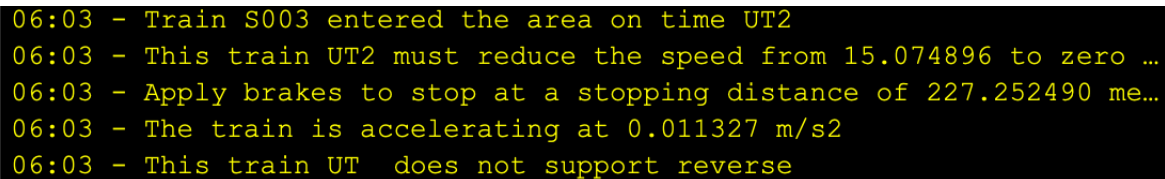
Create another track plan with UK Signal library and load it in TS2. Signals should be displayed on left side of the track plan.

Test cases are added to verify German Signal library .Test cases included loading of all the track items with desired length, next and previous route, max speed , signals to be displayed on the track and conditions. All the test cases are passed.

Constraint- TS2 editor has a bug which doesn't allow signals to change color automatically. Switching the scenery or zooming in/out of scenery triggers the signal update.

The implemented design is evaluated based on the changes in the existing system, Unit tests are done for the weight and engine power parameters as well as the reverse functionality of the trains is cross checked with the existing simulation traintypes below is a screenshot of trains that does not support reverse as well as the running trains which support reverse but need to reduce their speed from the current speed to zero. These are shown in the message logger for the user.

Along with these features braking distance is also displayed for the user in order to change the driving behavior of the user which may finally effect the energy consumption of the train.



```

06:03 - Train S003 entered the area on time UT2
06:03 - This train UT2 must reduce the speed from 15.074896 to zero ...
06:03 - Apply brakes to stop at a stopping distance of 227.252490 me...
06:03 - The train is accelerating at 0.011327 m/s2
06:03 - This train UT does not support reverse

```

Figure 11: Reverse directionality

For verifying the change in acceleration and deceleration values, possible values at different elapsed time frames are noted. But the change in acceleration and deceleration values must satisfy the constraints for energy conscious behavior i.e, for the existing project there is a rushed behavior of the train i.e, acceleration as quickly as possible with maximum jump in acceleration values at regular time intervals and applying brakes as slow as possible which is unsuitable for energy conscious, our project has the change in acceleration values with difference between the acceleration values as minimum as possible and also decelerating as quick as possible with difference between decelerated values as maximum as possible.

The speed profiles of both i.e, existing and updated projects are visualized at regular elapsed time frames precisely for every 2.5 seconds and the speed below is given in meter per seconds. Important aspect to be noted in the evaluation method is the increase in the constant speed timing as it consumes minimum energy when the train is accelerating at zero meter per second square, in our project the acceleration timing of constant speed is maintained for as maximum time as possible which finally consumes less energy.

$d := \text{math.Min}(0.5 * t.\text{TrainType}().\text{StdBraking} * \text{math.Pow}(\text{secs}, 2), 0.5 * \text{math.Pow}(\text{secs}, 2) * \text{stddec})$
 where d is the distance for stopping the train and figure 14 is the corresponding image.

The speed variations can be seen that the change in accelerated values is minimum with more constant speed in changed speed and due to applying brakes as quick as possible the train decelerates quickly resulting in further hike in acceleration in order to catch the speed towards the station. This non linear behavior is a backlog for the train and need to be improved considering the limit of decelerated values.

Unit tests are also performed on the speed at first elapsed time frame during the simulation which must match with the manually calculated values. Below is the detailed explanation of the manual calculation of the acceleration values.

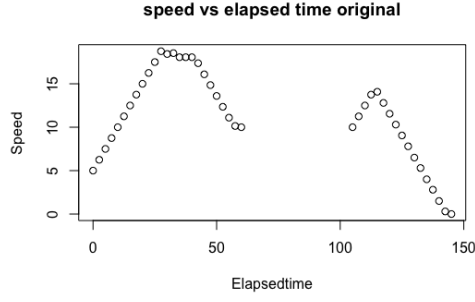


Figure 12: speed profile original

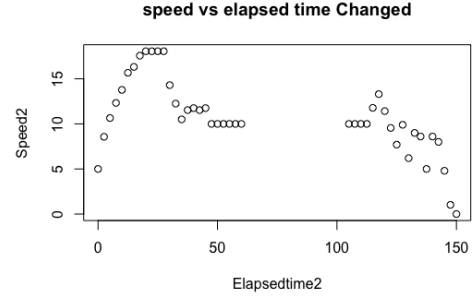


Figure 13: Speed Profile

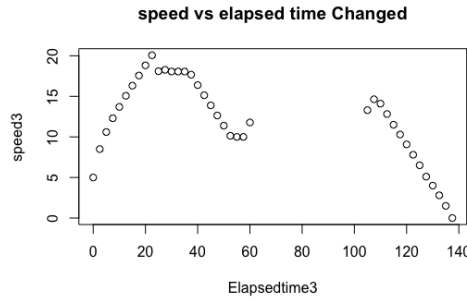


Figure 14: Speed Profile with d

```

1
2 acceleration := math.Max(-t.TrainType().EmergBraking,
3   math.Min(1/secs*(targetSpeed-t.Speed), math.Max(t.
4     TrainType().StdAccel,t.TrainType().Enginepower/(t.
5       Speed*t.TrainType().Weight)))
6 speed := t.Speed+acceleration*secs
7
8 calculate speed:=
9 math.Min(
10   getMaxSpeed(t),
11   math.Sqrt(math.Abs(2*targetDistance*t.TrainType().
12     StdBraking)
13   +math.Pow(targetSpeed, 2)))
14
15 getMaxSpeed :=
16 math.Min(t.TrainType().MaxSpeed, t.MaxSpeedForTrainTrackItems()
17 )
18
19 Values emergencybraking:= 1.5 , targetspeed := 25 ,
20 Stdaccel := 0.5 , enginepower := 5000 hp, Weight := 700

```

Initially target speed for the station is 5 i.e, initial speed and the target distance is zero i.e, initial distance .

Maxspeed of train := 25 , Maxspeed of trackitems := 27.7, secs :=2.5, targetdistance :=0

calculate speed gives the current needed speed

calculate speed := math.Min(getmaxspeed, sqrt(2*0*0.5 + 25*25))

```
getMaxspeed := Min(25, 27.7) :>25
```

```
acceleration := math.Max(-1.5, math.Min(1/2.5*(25-5),math.Max(0.5, 5000/(700*5))))
```

```
acceleration:= Max(-1.5, min(8, max(0.5,1.4285))) :> 1.4285
```

```
t.Speed := 5*1.4285*2.5 :> 8.571
```

These values calculated are matching with the simulated values.

Various trail and errors cases are used for observing the values in the min max functions across the time frames during the simulation. At any point of time during the simulation the train must not exceed the actual speed limit of the tracks as well as the max speed of the train. The train simulation has been checked and found out that it does not exceed the speed limit of 27.7 m/s and also as the latest simulation moves with low change in acceleration and at constant acceleration for long time the train arrives late than the normal time which clearly gives the picture of change in speed profile of the train.

All the code that has been modified and added is pushed to the git lab master branch where it must build , test and pass the pipelines with proper changes in order to merge through the existing project i.e, integrating with the existing project without any overlaps with other methods functionality. The code is successfully merged through the master branch by passing all the test cases previously defined as well as new cases added.

6 Conclusions

Existing Signal library was missing on German Signal library which has been designed and implemented. Most of the signals which German railways uses in real-time has been configured except SHUNT signals due to TS2 editor limitations.

As of now, signals are tested with simple track only as desired track loop was not implemented due to limited number of team-members. Due to which vertical placement of signal is also not implemented.

Overall the project realistic train engine movement has been improved with maximum features though there are still some limitations and further more performance improvement need to be done in terms of user inputs the project satisfies the current requirement of energy conscious behavior considering the factors of weight, engine power by controlling the rushed behavior of train and also reverse functionality of the train types has been set to specific trains. A design overview of the automated traffic controller and scoring system is an added advantage to the further developers.

There is lot more future scope in the project for instance adding of emergency brake button, providing the energy consumption of the train at any point of time in the simulation, implementing the regenerative braking for energy savings in e-trains, improving the speed profile with additional factors such as resistance, fuel type etc. Improving the user interface with lot more visibility of the running parameters during the train simulation.

Initial assumptions were made wrong with lot of possibilities which made to reconsider many scenarios of using the parameters at various contexts to achieve the goal. Trail and error methods using print statements during the simulation made to understand the code well for more improvements, the project has a lot more scope than expected to learn as well as to coordinate with other teammates for mutual transfer of knowledge.

All the tasks of project couldn't be accomplished as team size reduced from 5 to 2. Initial tasks assigned to team-members are kept open as team-members dropped from the project in between and that effected our end goal. Leaving the project towards the end impacted our project plans a lot as there was not enough time to redistribute the tasks.

However, whatever possible has been achieved by rest of the team-members by regular follow-up meeting and better co ordinations.

References

- [1] Nguyen, N. T. BahnDSL: A Domain-Specific Language for Configuring and Controlling Railways. Master's thesis, The University of Bamberg, Germany, WiSe 2019/20.
- [2] Physics of train acceleration
- [3] Chul-Ho Kim, Kee-Man Lee Analytical Study on the Performance Analysis of PowerTrain System of an Electric Vehicle, World Electric Vehicle Journal Vol. 3 - ISSN 2032-6653 - © 2009 AVERE.
- [4] Brooklynrail science of railway locomotion
- [5] Velocity acceleration time distance
- [6] Deng Pan, Qing Luo, Liting Zhao, Chuansheng Zhang, Zejun Chen, "A New Calibration Method for the Real-Time Calculation of Dynamic Safety Following Distance under Railway Moving Block System", Mathematical Problems in Engineering, vol. 2018, Article ID 3061034, 11 pages, 2018. <https://doi.org/10.1155/2018/3061034>
- [7] Energy efficient driving
- [8] Recuperation of braking
- [9] Skrucany, Tomas, Vrabel, Jan and Kazimir, Patrik. "The influence of the cargo weight and its position on the braking characteristics of light commercial vehicles" Open Engineering, vol. 10, no. 1, 2019, pp. 154-165. <https://doi.org/10.1515/eng-2020-0024>
- [10] Standard deceleration of the train
- [11] Swtbahn-Signals
- [12] Hauptsignal/Vorsignal
- [13] TS2 Technical Manual
- [14] Optimal train movement for minimum energy consumption

A Sprint Goals

Sprint 1. *Distribution of tasks, creating sprint planning and issues.* Setup and Complete Understanding of the project.

Sprint 2. *Understanding existing Signal behaviour and TS2 editor functionality.* Creating signal designs Implementation of Directionality and study of weights.

Sprint 3. Implementation of German Signal aspects and signal types Creating track loop and integrating German signals Study of Energy Conscious and implementation of weights

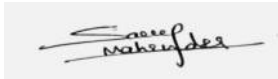
Sprint 4. *Distribution of tasks, creating sprint planning and issues.* Extending German signal library with Distant signals Research study and documentation about SHUNT signals Debugging and change in Speed profile

Sprint 5. *Distribution of tasks, creating sprint planning and issues.* Testing of implemented German Signal library Creating project and lab reports Research about Automated traffic controller Controlling rushed Behavior and writing testcases

Ehrenwörtliche Erklärung

Alle Unterzeichner erklären hiermit, dass sie die vorliegende Arbeit (bestehend aus dem Projektbericht sowie den separat abgelieferten digitalen Werkbestandteilen) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

1987471	<i>Deepika Arneja</i>
Matrikelnummer	Name
Bamberg, 15-02-2021	<i>Deepika arneja</i>
Ort, Datum	Unterschrift

1985741	<i>Mahender reddy Sheri</i>
Matrikelnummer	Name
Bamberg, 15-02-2021	
Ort, Datum	Unterschrift