Høgskulen på Vestlandet

Bacheloroppgave

ELE350

Predefinert informasjon

Startdato:	08-05-2023 09:00 CEST	**Termin:**	2023 VÅR
Sluttdato:	22-05-2023 14:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
Flowkode:	203 ELE350 1 O 2023 VÅR		
Intern sensor:	Ingvar Henne		

Deltaker

Navn:	Benjamin August Lindbæck
Kandidatnr.:	260
HVL-id:	585125@hvl.no

Informasjon fra deltaker

Egenerklæring *:	Ja
Inneholder besvarelsen konfidensielt materiale?:	Nei
Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *:	Ja

Gruppe

Gruppenavn:	BO23EB-35 Fagskole Modelljernbane
Gruppenummer:	28
Andre medlemmer i gruppen:	Jakub Grzybowski, Emil Fagerli Osvoll

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

Er bacheloroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Western Norway University of Applied Sciences

Bachelor Thesis

Model Railway Control using Collaborating Processing Units

Benjamin August Lindbæck
Emil Fagerli Osvoll
Jakub Grzybowski

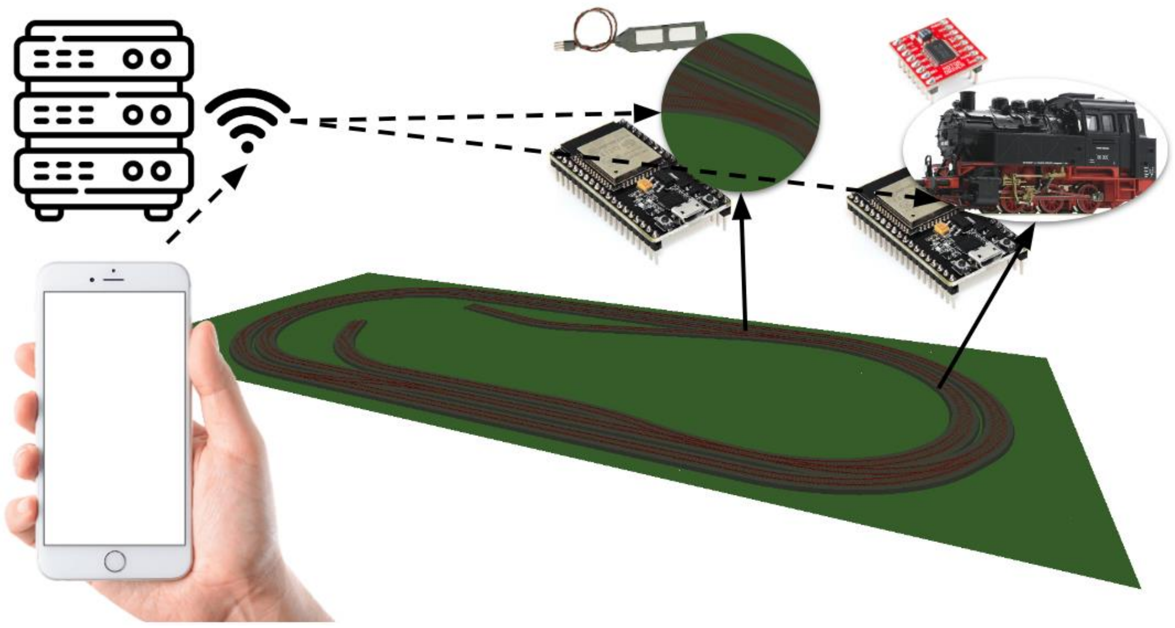
22. May. 2023

Publishing Agreement

Report Title: Model Railway Control with Collaborating Processing Units	*Date/Version* 22. May. 2023/1.1
	Report number: BO23EB-35
Authors: Benjamin August Lindbæck Emil Fagerli Osvoll Jakub Grzybowski	*Course:* AUTB20
	Number of pages including appendices: 63
Western Norway University of Applied Sciences Tutor: Ingvar Henne	*Grading:* Open
Comments: We, the authors, allow the report to be published.	

Client: Fagskulen Vestland	*Client reference:*
Client contact person: *Guttorm Lyngvær*	

Jakub Grzybowski	Emil F. Osvoll	Benjamin lindbæck
Jakub Grzybowski	Emil Fagerli Osvoll	Benjamin Lindbæck



Preface

This report presents research performed on the behalf of Fagskulen Vestland, conducted as students in the course "Automation with Robotics" at Western Norway University of Applied Sciences, campus Bergen. The research project was undertaken as part of the subject "ELE350-1 Bacheloroppgave", with the objective of implementing a model railway control system using collaborating processive units.

We would like to express our sincere gratitude to our internal supervisor, Ingvar Henne, for his invaluable guidance and support throughout the duration of this project. His expertise in digital systems and industry 4.0 proved instrumental in identifying areas for improvement and providing valuable insights.

Furthermore, we extend our appreciation to our external supervisor, Guttorm Lyngvær, for his assistance in clarifying the project requirements and for providing us with access to facilities, tools, and components necessary for our research.

This report aims to contribute to the field of "Automation with Robotics" by presenting our analyses and findings. It is our hope that this research will enhance our understanding of model railway control systems and provide practical insights regarding the implementation of collaborating processing units.

Summary

This bachelor thesis discusses the development of a model railway control system using collaborating processing units. By implementing carefully selected actuators and sensors, basic functionalities such as the movement of trains, track switches and traffic lights can be controlled and monitored by an operator. The control interface is accessible through a locally hosted website, thereby allowing users the chance to control the system using a mobile device or similar.

To establish connectivity between the user and the model railway, some core concepts from industry 4.0 are implemented, such as IoT and M2M. ESP8266 and ESP32 microprocessors with built-in wireless transmitters are connected to the sensors and actuators, enabling them to communicate with a Ubuntu server. The server achieves wireless connectivity with the use of a regular Wi-Fi router. And by hosting a Mosquitto broker, an MQTT bridge, a web server and an API, the collaboration between processing units is facilitated.

The objective of incorporating a solution to control the model railway system using digital network devices was mostly achieved. However, difficulties were encountered when implementing the track switching actuators. And although the user interface to control the model railway achieved desired functionality, there is still room to develop a more sophisticated monitoring interface.

Sammendrag

I denne bacheloroppgaven blir utviklingen av et modelljernbanekontrollsystem som tar i bruk samarbeidende prosesseringsenheter utforsket. Ved å anvende nøye utvalgte aktuatorer og sensorer, så kan grunnleggende modelljernbanestyring av tog, sporvekslere og trafikklys overvåkes og kontrolleres av en operatør. Grensesnittet som gjør det mulig for brukeren å styre systemet er tilrettelagt ved bruk av en lokalt vertet nettside, noe som gjør at det er mulig å styre modelljernbanesystemet ved bruk av nettverksenheter som mobiltelefoner eller lignende.

For å tilrettelegge for at brukerens enhet kan kobles opp mot modelljernbanens styringsmekanismer, så blir noen sentrale aspekt ved industri 4.0 som IoT og M2M benyttet. ESP8266 og ESP32 mikroprosessorer med innebygde trådløse antenner er koblet opp mot sensorer og aktuatorer, noe som gjør at data kan overføres mellom mikroprosessorene og en Ubuntu-server. Serveren oppnår denne trådløse nettverkstilkoblingen ved å benytte en helt vanlig Wi-Fi nettverksruter. Og ved å verte en Mosquitto-megler, en MQTT-bro, en webserver og et API, så tilrettelegges det for at de ulike prosesseringsenhetene kan samarbeide.

Utfordringen med å utvikle en løsning som tilrettelegger for modelljernbanestyring over digitale nettverksenheter ble, for det meste, oppfylt. Det oppstod riktig nok utfordringer når de tiltenkte sporveksleraktuatorene skulle styres. Og selv om brukergrensesnittet vi utviklet oppnådde den tiltenkte funksjonaliteten til å styre modelljernbanesystemet, så er det fremdeles rom for å utarbeide et bedre overvåkingsgrensesnitt.

1 Table of Contents

List of Abbreviations

3D	Three-dimensional
API	Application programming interface
CPU	Central processing unit
DC	Direct Current
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
I/O	Input and output
IoT	Internet of things
IR	Infrared
JSON	JavaScript object notation
LED	Light emitting diode
M2M	Machine to machine
MQTT	Message Queuing Telemetry Transport
PCB	Printed circuit board
PLA	Polylactic acid
PWM	Pulse width modulation
SDK	Software development kit
UI	User interface

This report adheres to the SI (International System of Units) measurement standard.

2 Introduction

2.1 Client

The client, Fagskulen Vestland, has issued five different bachelor thesis projects to the "Automation with Robotics" course at Western Norway University of Applied Sciences in Bergen. This bachelor thesis aims to address one of the specified challenges. Fagskulen Vestland is one of the largest vocational schools of its kind in Norway [1]. The department responsible for this project is in Nordnes, Bergen, and offers a variety of technical- and health-related studies.

2.2 Requirements

- System Architecture Design:
    a. The control system should allow users to operate track switches, traffic lights and control the speed and direction of model trains on the multi-train model railway.
    b. The control system architecture must incorporate a carefully selected set of actuators, sensors, and other relevant components to meet the specific requirements of the existing model railway set.
    c. The system must embrace the principles of Industry 4.0, integrating cooperative processing units that enable seamless collaboration between various components.
- Digitalization of Existing Model Railway Set:
    a. The control system must be adaptable to the pre-existing RocoLine model railway set.
    b. The chosen approach should ensure compatibility and successful integration of the control system with the existing model.
- Wireless Control and Communication:
    a. The proposed control system should enable users to wirelessly control the model railway using mobile devices.

2.3 Problem Description

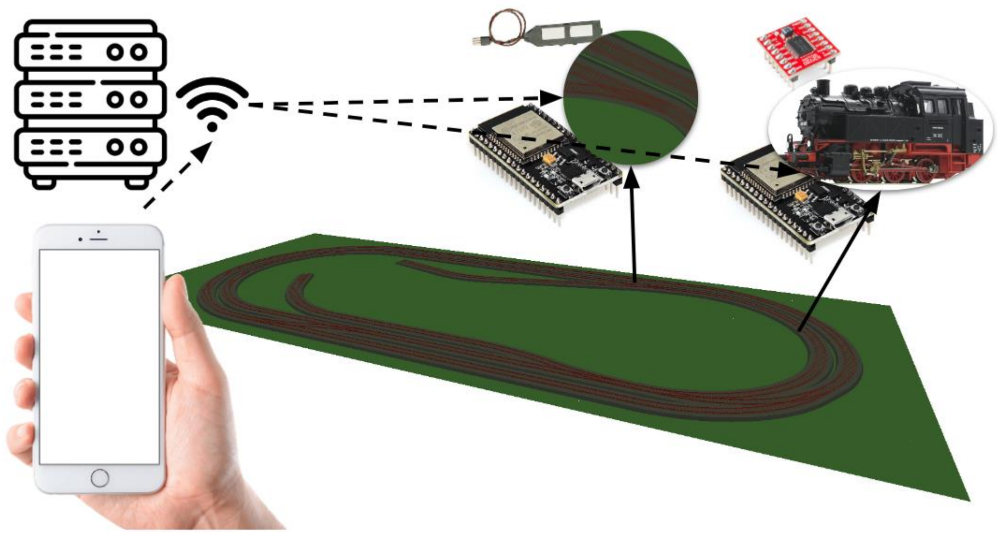


Figure 1-1

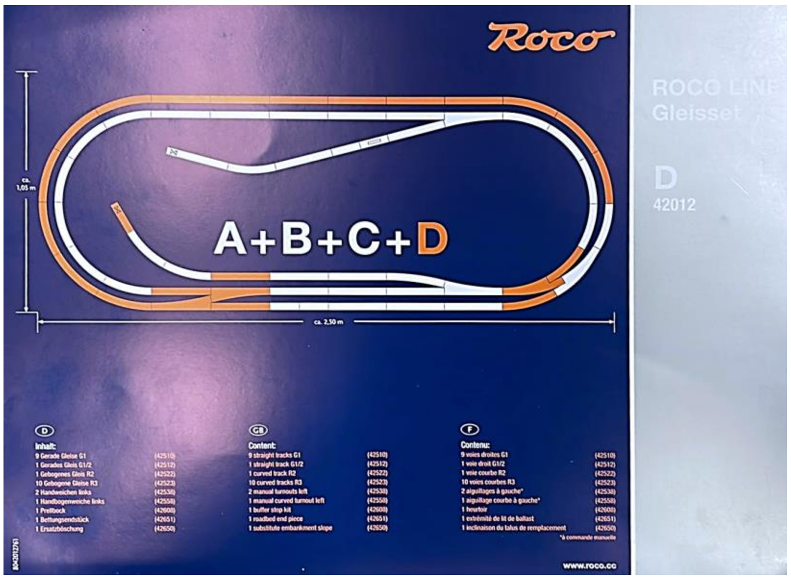
The objective of this bachelor project is to develop a digital control system for a multi-train model railway. This must allow users to digitally operate track switches, traffic lights, and control the speed and direction of model trains. The actuators, sensors, and other relevant components used to facilitate this digitalized solution must be carefully selected to integrate with a pre-purchased model railway kit from “ROCO Line.”

The challenge of this project is to devise a solution incorporating the concept of collaborating processing units and other relevant concepts from industry 4.0. To accomplish this, we must thoroughly research the model railway components. The desired outcome is a digitally controlled system that focuses on M2M communication and cooperative processing units. By approaching the project this way, our goal is to showcase a practical implementation of industry 4.0 concepts and technologies.

3 Analysis of the Problem and Implementation

3.1 Model Railway Kit

To start the project, the process of assembling the model railway kit that the client provided began. Scaled 1:87 relative to their real train counterparts[2], the components are manufactured by the Salzburg-based[3] company "Roco". The client instructed that tracks are to be configured as displayed in *figure 2-1*. A configuration that, under the article number "ROC42012", is titled "RocoLine Gleisset D". However, the tracks, as lined up with the instructions, did not match the diagram. And so, a decision was made to change the track configuration to resemble the diagram more closely. This was successfully implemented.



Ⓓ

Inhalt:	
9 Gerade Gleise G1	(42510)
1 Gerades Gleis G1/2	(42512)
1 Gebogenes Gleis R2	(42522)
10 Gebogene Gleise R3	(42523)
2 Handweichen links	(42538)
1 Handbogenweiche links	(42558)
1 Prellbock	(42608)
1 Bettungsendstück	(42651)
1 Ersatzböschung	(42650)

GB

Content:	
9 straight tracks G1	(42510)
1 straight track G1/2	(42512)
1 curved track R2	(42522)
10 curved tracks R3	(42523)
2 manual turnouts left	(42538)
1 manual curved turnout left	(42558)
1 buffer stop kit	(42608)
1 roadbed end piece	(42651)
1 substitute embankment slope	(42650)

Ⓕ

Contenu:	
9 voies droites G1	(42510)
1 voie droit G1/2	(42512)
1 voie courbe R2	(42522)
10 voies courbes R3	(42523)
2 aiguillages à gauche*	(42538)
1 aiguillage courbe à gauche*	(42558)
1 heurtoir	(42608)
1 extrémité de lit de ballast	(42651)
1 inclinaison du talus de remplacement	(42650)

*à commande manuelle

Figure 2-1

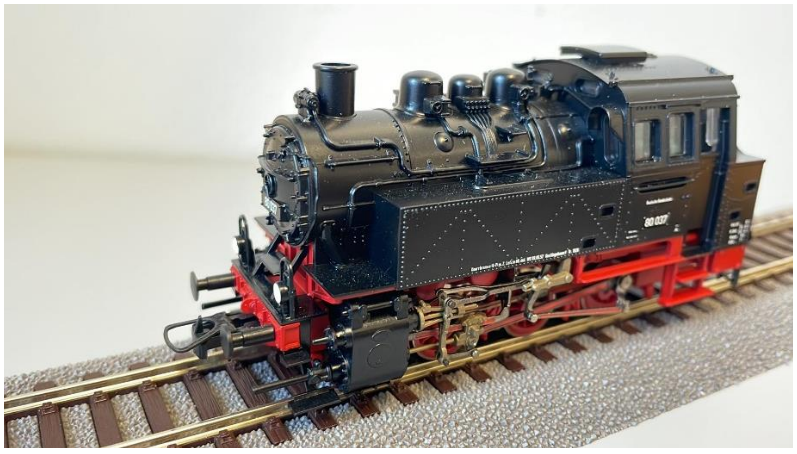


Figure 2-2

3.2 Support Structure

The use of a support structure to raise the model railway track assembly off the floor offers several practical advantages. Firstly, it positions the model railway at a comfortable working height, which, in this case, is set at waist level. Additionally, a support structure enables the convenient installation of wires, actuators, sensors, and other similar components. To build this structure, our group was supplied with a limited amount of sheet wood and two-by-four dimensional lumber from Fagskulen Vestland.

Taking these constraints into account, a support structure measuring approximately 96 centimeters in height, 240 centimeters in length, and 120 centimeters in width has been constructed. To facilitate the installation of an electrical control panel, an attachment plate was installed underneath the tabletop.

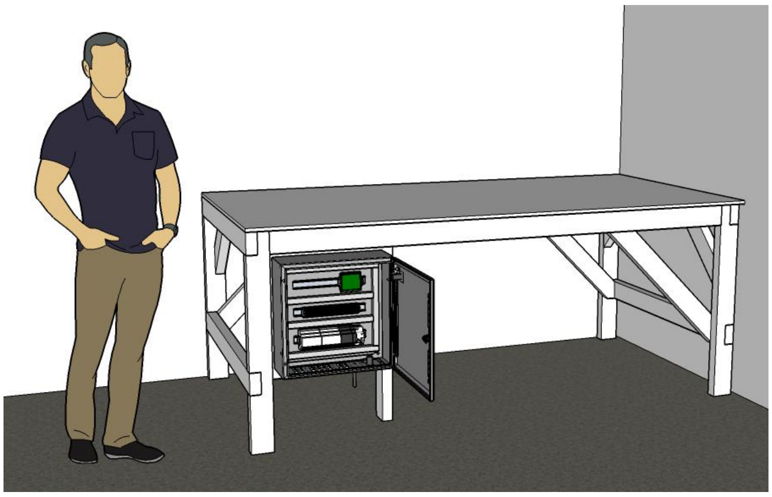


Figure 2-3

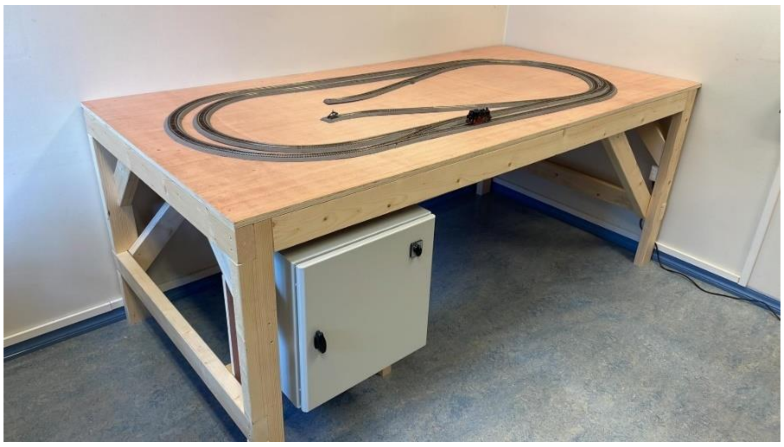


Figure 2-4

3.3 Actuators

An actuator is a device that produces a motion by converting energy and signals going into the system. The motion it produces can be either rotary or linear.[4]

3.3.1 Train Movement

Train movement can be understood as the direction in which a train moves at a certain velocity. As specified in the problem description, a solution to control this digitally must be developed. To address this challenge, the client was questioned about their requirements. They provided clarity on two important concerns: How many trains were meant to be controlled simultaneously and what type of modifications could be made to the components of the model railway kit. The answers they provided implied that it must be possible to control multiple trains simultaneously. Also, the internal mechanisms of the trains could not be altered.

With these criteria in mind, the challenge is divided into several smaller challenges. The first one is to understand what type of propulsion system must be controlled, as different propulsion systems may require remarkably different solutions. Researching the components established that the trains are propelled electrically. Specifically, by internal direct current motors. And when controlling the voltage across the two poles of a direct current motor, it is possible to control both the rotational speed and direction of the output shaft. In this case, the rotational motion is transferred to the wheels of the train, which in turn is proportionally translated into a linear motion along the railway tracks. For the sake of simplicity, factors such as friction are negated, as they are perceived to minimally affect the functionality of this system.

The next challenge is to address how electrical energy can be supplied to these motors. It is common to see electrical devices supplied with electricity via an internal or external source. However, as it is not permitted to alter the trains internally, solutions such as batteries can be eliminated. External sources are normally connected with cables. But because trains are inherently dynamic objects, tethering them with cables would limit their mobility and is hence eliminated as a viable solution. However, both the trains and the train tracks have electrical conductors that facilitate energy transfer from an external source. And so, by using one track as a negative pole and the other as a positive pole, a train will, when placed upon the tracks, conduct the direct current to each pole of its motor.

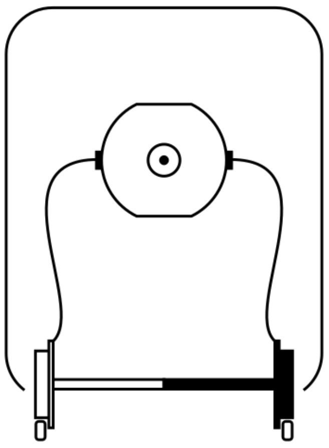


Figure 2-5[5]

With the suggested solution, another challenge is introduced. If one considers multiple trains on top of the tracks in this configuration, they will always be energized equally. Therefore, the trains cannot be controlled individually as is. So how can one work with this idea and still achieve a perceived state of individuality? To address this, it was decided to isolate the model railway tracks into six individually controlled sections. This is a common solution to the problem and is achieved with specialized isolator components[5] included with the model railway kit. By doing so, a user can practically operate multiple trains at the same time. But as you introduce more than six trains to this system, it is theoretically impossible to control all of them individually. However, because the client deemed this satisfactory, it was implemented as a solution.

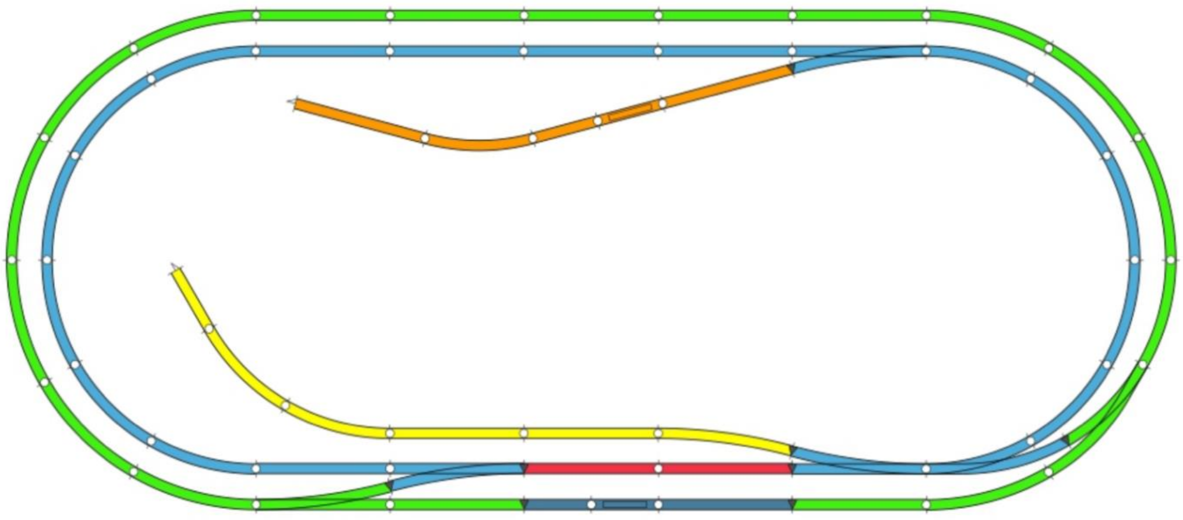


Figure 2-6

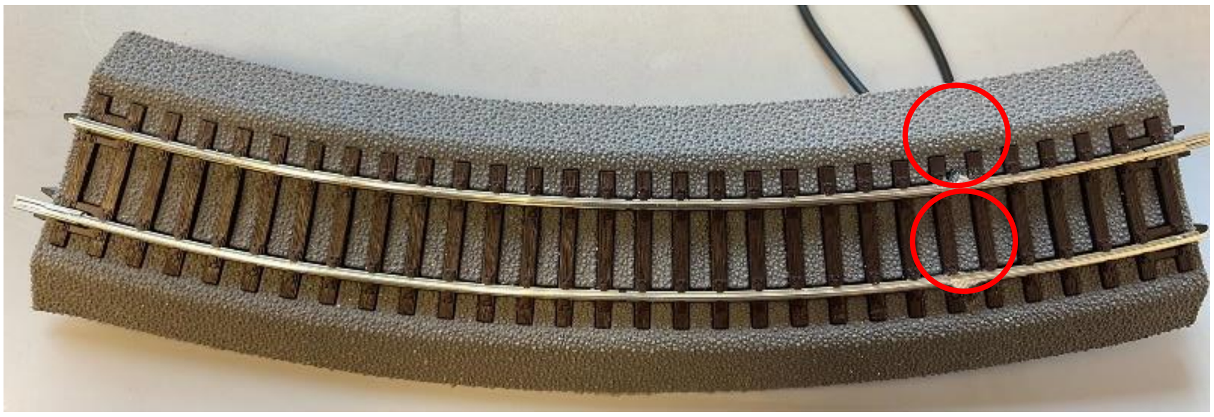


Figure 2-7: To connect these rail sections to an external source of electricity, a wire was soldered

3.3.1.1 Motor Drivers

As mentioned in the introduction, this actuating system is meant to be controlled digitally. However, the manufacturer intended for these motors to be controlled with analog controllers. Therefore, the next challenge was to devise a solution that can meaningfully convert digital signals into analog signals. It is quickly established that components called "motor drivers" are suited to the purpose. And for a motor driver to suit this specific system, it is important that it can control direct current motors while functioning within the voltage range of zero to fourteen volts.

Motor Driver	**L298N**[6]	**TB6612FNG**[7]
Input voltage range [V]	5	2.7 – 5.5
Output voltage range [V]	5 – 35	4.5 – 13.5
Number of motors per drive	2	2

Two options for motor drivers were considered: The L298N and the TB6612FNG. The L298N is limited to five-volt digital control signals. This makes it an unsuited candidate when one takes into consideration the selection of signal processing devices, which will be discussed in detail later. The L298N also incorporates a heat sink due to its efficiency of only 40 to 70%. In comparison, the TB6612FNG does not need a heat sink because of its efficiency of 90%.[8] Therefore, the TB6612FNG is physically smaller than the L298N. With these benefits, and with suitable voltage operating ranges, the TB6612FNG was selected for converting digital control signals into analog motor driver signals.

With six different track systems to control, three motor drivers would be adequate. However, the client stated that the model railway system might be expanded in the future. To address this, it was decided to incorporate six motor drivers so that upwards of twelve different train sections can be controlled.

3.3.1.2 Printed Circuit Board

Having 16 pins each, six motor drivers would give a combined number of 96 pins to wire. It is possible to wire all these pins with individual strands. However, it is not a very elegant solution within the confines of a small junction box. Therefore, it was deemed appropriate to design a printed circuit board, hereby abbreviated to "PCB." Another advantage with a PCB is that it eases installation and maintenance as it serves to reduce the number of input and output pins. With these objectives in mind, the process of designing it was initiated.

First, it was discovered that to set the direction of a motor with the TB6612FNG, one must provide signaling to two pins. And for the motor to rotate, one of them must be set to a high state, while the other is set to a low state. This logic can be simplified with a NOT-gate so that the motor direction is controlled with only one input pin. Specifically, the SN74HC04N was selected for this purpose. Along with other simplifications such as permanently setting the standby pin to a high state, the motor driver inputs were simplified so that only two pins are needed to control each motor. Later, when the development of a control system is discussed, one must be aware that one pin controls the speed, and the other controls the direction. The speed-pin takes a pulse width modulated signal, and the direction-pin takes a Boolean signal.

Then, to prevent the wire traces within the PCB from crossing each other, it was decided to use a four-layer stack, where the two outermost layers are solid ground planes. Another confinement that had to be taken into consideration was the size of the junction box that would house the board. Therefore, the board dimensions were carefully selected.

To translate the proposed ideas into a physical PCB, the digital circuit design software "KiCad 6.0" was utilized. After finalizing the design, the last step before manufacturing was to compile a Gerber-file. It contains all the necessary data that a manufacturer needs to produce a PCB. In this instance, "JLCPCB" was the vendor selected to manufacture it.

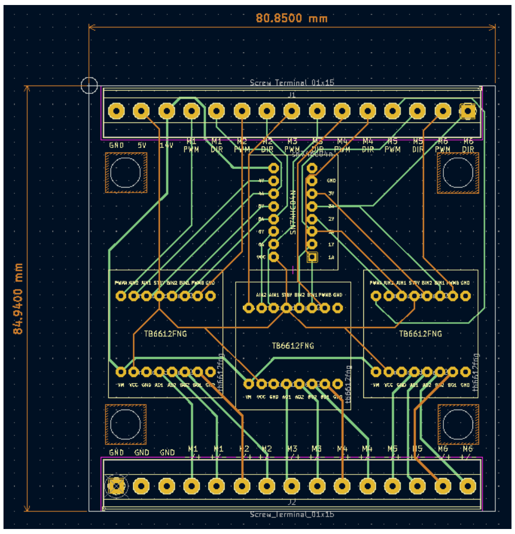


Figure 2-8

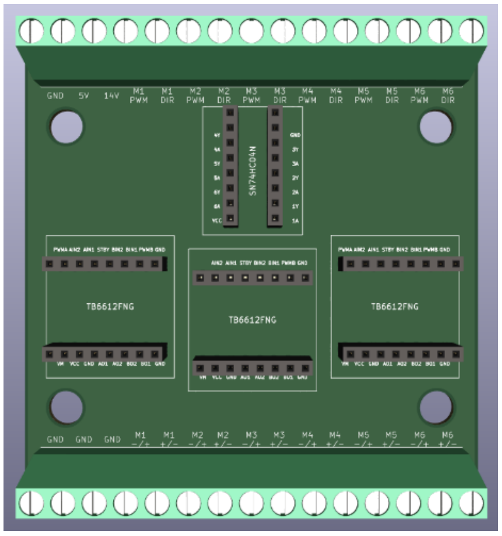


Figure 2-9

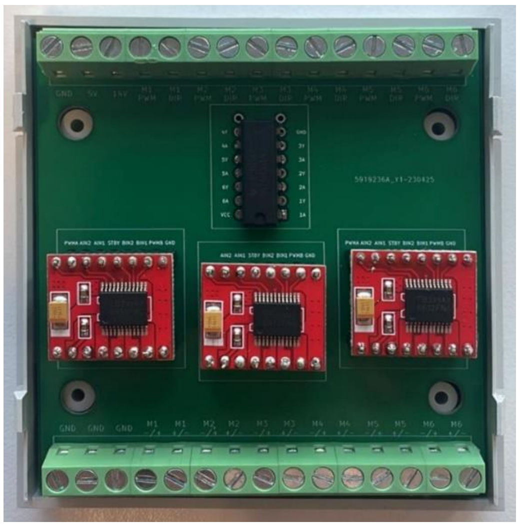


Figure 2-10

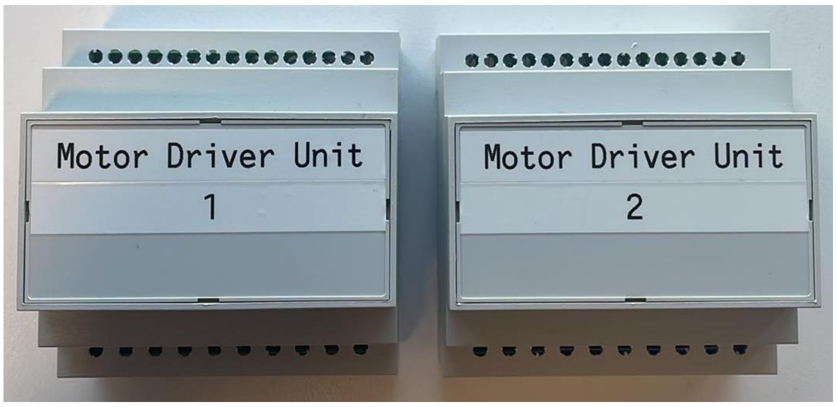


Figure 2-11

3.3.2 Track Switches

The purpose of a track switch is to let trains move between different train tracks. As established in the problem description, a solution to control track switches must be incorporated and it must function with this specific model railway system. That eliminates the need to design actuating solutions for anything other than two-way track switches.

So, to address this challenge, research was initiated to understand the mechanisms involved. For demonstration purposes, a ROCO WR-15 is displayed in figure 2-12. As seen above the blue circle labeled 1, an arrow pointing to the left and to the right refers to the movement of a small lever that exists on each track switch. When manipulated by hand, the track switch can be set to its desired state. The two possible states are visualized by the lines next to the second label. The most important discovery is that, unlike the trains, these track switches have no pre-existing internal actuators because they are meant to be operated by hand.

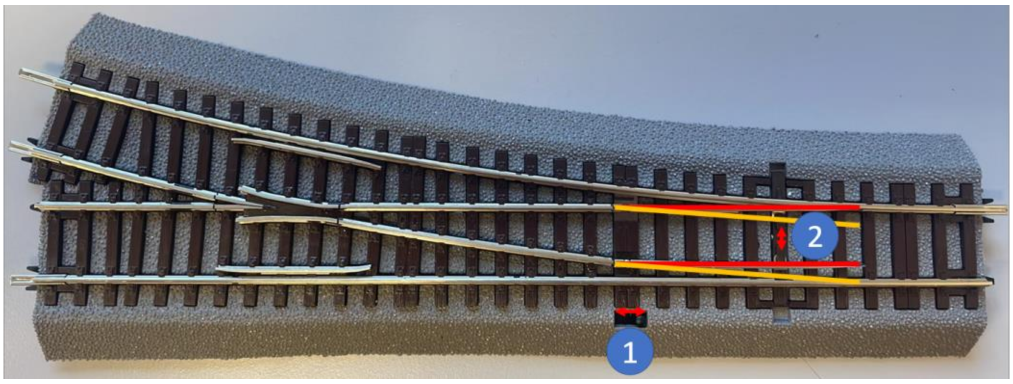


Figure 2-12

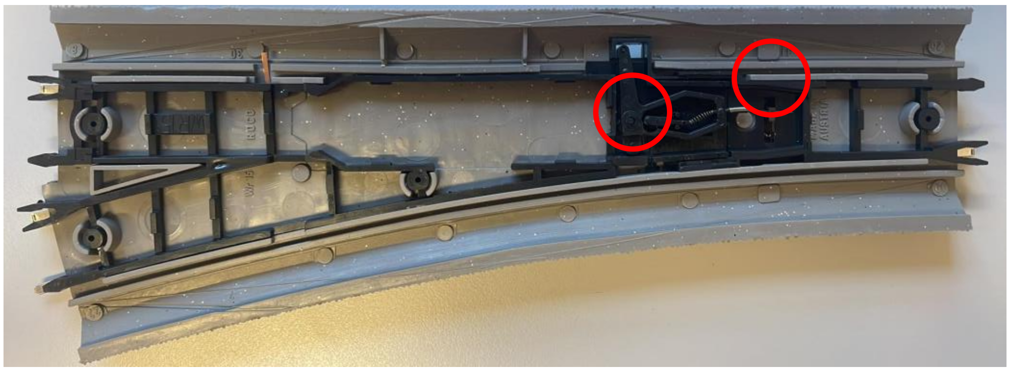


Figure 2-13

And so, the ROCO WR-15 was turned upside down with the purpose of revealing ways to connect an actuator. After removing a protective cover, two potential attachment points were discovered, as encircled in figure 2-13. What they both have in common is that they move linearly. And the distance that they move is about 2.5 millimeters. Therefore, it would be ideal to incorporate an actuator that produces a linear motion. Although it is possible to convert rotational motion into linear motion, it is impractical, and so rotational actuators such as servos are not deemed viable. Solenoids are known to produce a linear motion while moving between two states, just like track switches do. Because of this, a solenoid was selected to actuate the system. Fortunately, ROCO has already designed a solenoid that can be implemented with these specific track switches. They use the leftmost attachment point displayed in figure 2-12. and 2-13 so-13. So, with six different track switches, six of these ROCO 42620 solenoids were purchased.

The next challenge to address was how these solenoids can be manipulated. From reading the manual [9], it was understood that the manufacturer intends for them to be installed with a proprietary control system. However, using an already existing control system solution would defeat the purpose of this bachelor thesis, which aims to develop such solutions from scratch. And so, to understand how the solenoids can be actuated, it was discovered that the three wires attached to each solenoid, as displayed in figure 2-14, are two signaling wires and one grounding wire. By connecting the black wire to ground and setting the red and green wire to a high and low state, the actuator could be controlled. Just like the motor driver’s direction pins, the red and green wires are always in opposite states. And by selecting the order, one can control the state of the track switch. The manual suggested that a high state lies between fourteen to sixteen volts DC. A low state is achieved when the wire is grounded.

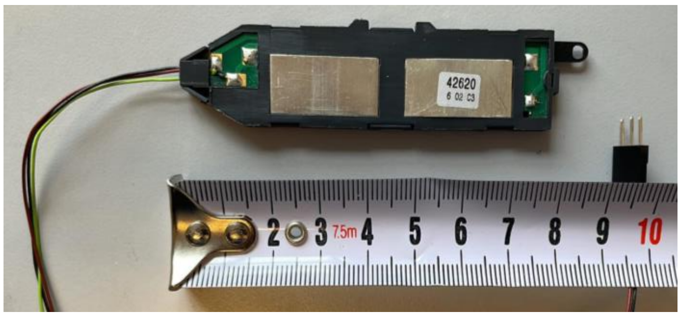


Figure 2-14

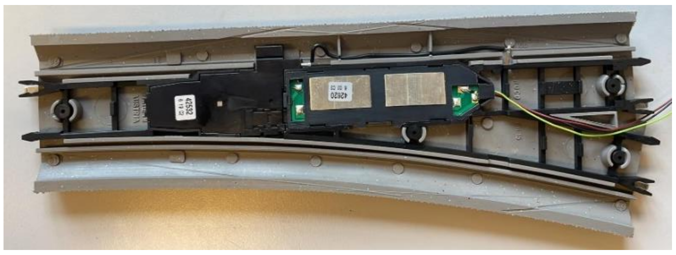


Figure 2-15

3.3.3 Traffic Lights

The intention of traffic lights is to provide a visual cue to the train operator that some type of operation of the train is to be initiated. And for them to serve a meaningful purpose, they will be utilized to prevent trains from colliding. Such incidents could occur when they move along the same track towards each other, or when they cross paths through intersections. A choice is made to concentrate the traffic lights around intersections. This is because collisions on the long straight sections are considered highly unlikely with the configuration of six independently controlled track sections, as referred to in the “Train Movement”-section.

To develop a solution that effectively communicates the intended two states of operation, red and green are chosen as appropriate colors: Green for signaling that a train can move ahead, red for signaling that it cannot. Although there is an argument to be made about green and red being unsuited colors due to the prevalence of color-blindness [10], they are commonly used with traffic lights [11]. Therefore, it is assumed that many people are likely to interpret the color scheme correctly without the need to be informed about it explicitly.

A logic that determines the state of each traffic light must be established. And because the problem description provides few constraints, a decision is made to keep it as simple as possible. One alternative is to connect the logic of the traffic lights to the logic of the track switches. However, this introduces the challenge of mapping the right signals to the correct lights. And to propose a solution, it is useful to first map out the three relevant intersections with a suggested traffic light configuration. Such a proposition is suggested in the three following figures, figure 2-16, 2-17, and 2-18. The arrows at the top of a light assembly indicate the direction in which it applies. Also, the red arrows are used to indicate where a traffic light should be installed. At these proposed locations, traffic lights face towards trains moving into the nearest intersection.

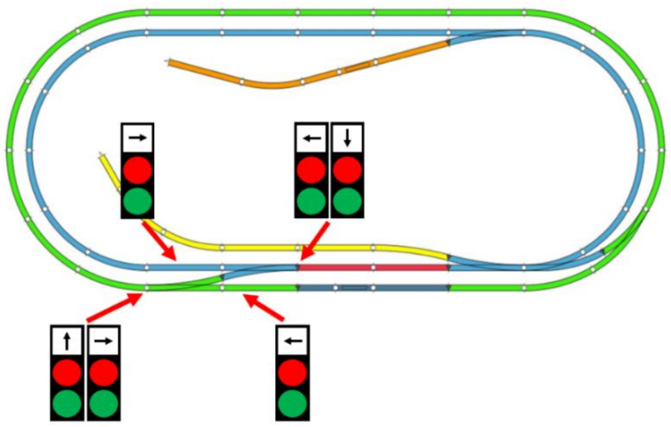


Figure 2-16

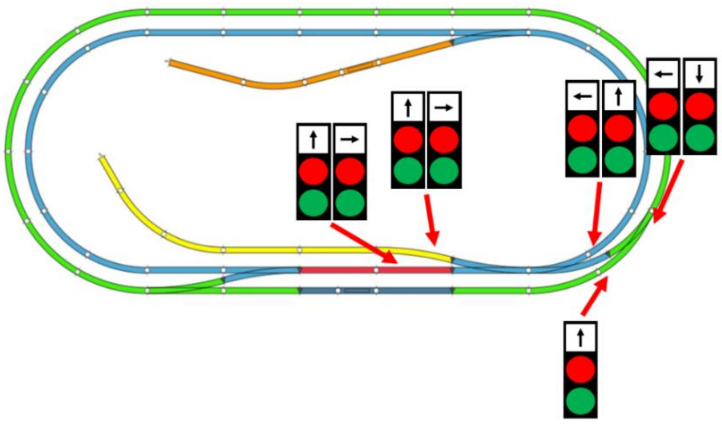


Figure 2-17

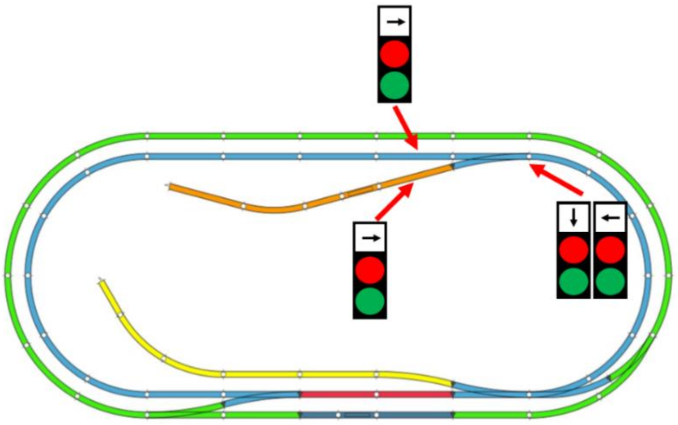


Figure 2-18

Because the logic applied to the traffic lights stems from the track switch actuation, it must be configured sensibly. And so, a green light can only be given if the track switches are configured to allow movement in the direction that a traffic light refers to. In any other situation, a red light will be signaled. However, this introduces a limitation that must be considered. If two trains move towards the same intersection from opposite directions, the lights will not indicate that they are about to collide into each other. So, there is considerable room for improvement to this logic when prioritizing the development of collision avoidance systems. However, the problem description does not put an emphasis on that type of system, and so this solution is deemed adequate.

To create a physical implementation, components were designed in the digital 3D-modelling software «Trimble Sketchup». The final design incorporates one red and one green common LED for each assembly. After finalizing the design, the black pieces, as displayed in figure 2-19, 2-20, 2-21, and 2-22, were sliced using «FlashPrint 5». This facilitated the production of 19 traffic light assemblies with a FlashForge Finder 2.0 3D-printer extruding black PLA-filament. Then, to join the bottom and top section, round wooden barbeque skewers were cut to length and inserted into the holes visible in figure 2-22. The arrows, as displayed in figure 2-19, figure 2-16 to 2-20, and figure 2-21-18, were printed with a label maker and adhered to the barbeque skewers. The arrows, as displayed in figure 2-19, 2-20, and 2-21, were printed with a label maker, and adhered to the barbeque skewers.

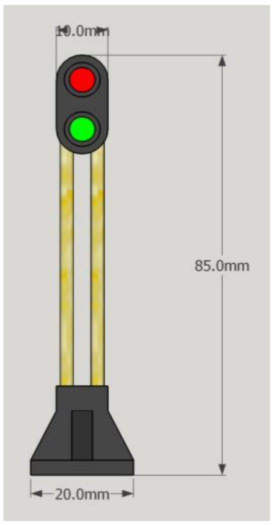


Figure 2-19

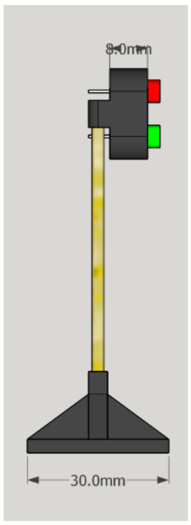


Figure 2-20

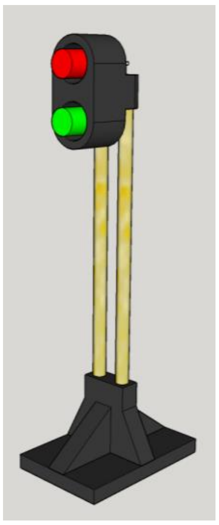


Figure 2-21

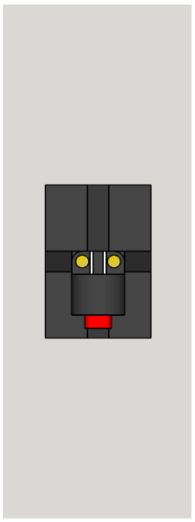


Figure 2-22

3.4 Sensors

A sensor is a device that detects and responds to some type of input from the physical environment. The input can be light, heat, motion, moisture, pressure, or any number of other environmental phenomena. The output is generally a signal that is converted to a human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.[12]

3.4.1 Train Localization

The problem description does not explicitly demand that a solution for localizing trains must be implemented. And as already discussed under "Traffic Lights," traffic collision avoidance systems are not prioritized as a part of this thesis. However, the client has communicated that such solutions might be implemented in the future. Therefore, a decision was made to incorporate components that facilitate the development of a control system related to the movement of the trains.

When discussing control systems in general, it is useful to establish whether they are open-loop systems or closed-loop systems.

An Open-loop system, also referred to as non-feedback system, is a type of continuous control system in which the output has no influence or effect on the control action of the input signal.[13]

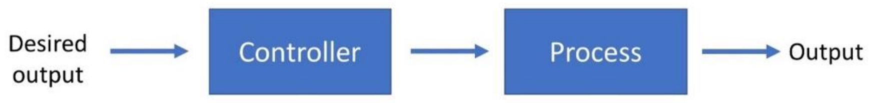


Figure 2-23[14]

A Closed-loop Control System, also known as a feedback control system is a control system which uses the concept of an open loop system as its forward path but has one or more feedback loops (hence its name) or paths between its output and its input.[14]

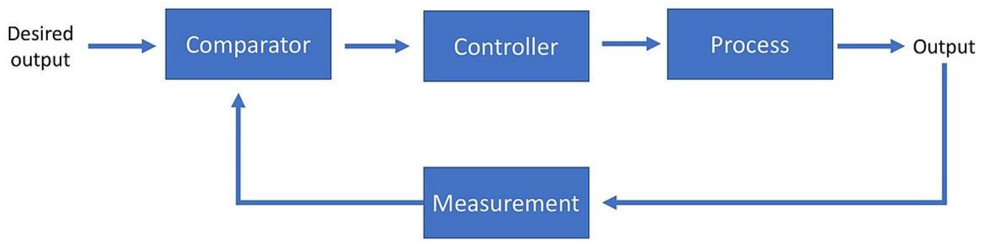


Figure 2-24[15]

Automatic train collision avoidance systems are inherently closed-loop systems as they require feedback about the trains speed, direction, and localization to prevent them from colliding. As referred to in “Train Movement,” each track section is controlled to a set speed and direction. And so, by identifying what track section a train is located upon, it is possible to approximate the speed and direction of it. Hence, both the speed and the direction are functions of the position. This eliminates the need to incorporate sensors for all three variables. Only sensors that provide localization feedback are considered.

The next step is to evaluate potential solutions for providing localization feedback. And for it to be suited, resolution must be accommodated for.

Resolution is the smallest possible change that a sensor can perceive.[16]

As discussed under the “Traffic Lights”-section, collisions are deemed most likely to occur around intersections. For this reason, the resolution of the feedback should be higher in these areas when compared to the long, straight sections. If the sensors provide localization feedback in time for the closed loop control system to make necessary corrections that prevent collisions, the resolution is deemed adequate. To quantify the desired resolution, tests had to be performed on the movement of the trains. It revealed that a train can perform a controlled stop, from full speed to a complete standstill, over approximately 20 centimeters. Therefore, the trains must be localized, at latest, before they are within 20 centimeters from an intersection. To add a margin of safety, they should be localized as early as approximately 40 centimeters, or more, before the intersection.

Having established requirements for the resolution, it is also ideal to select solutions that ease implementation while remaining affordable. For this reason, candidates such as motion detecting cameras or similar were not considered. Also, to keep the model railway assembly as tidy as possible, a decision was made to incorporate sensors underneath the train tracks. Hall effect sensors and infrared proximity sensors were considered potential candidates within this set of criteria.

The hall effect sensor is a type of magnetic sensor which can be used for detecting the strength and direction of a magnetic field produced from a permanent magnet or an electromagnet with its output varying in proportion to the strength of the magnetic field being detected.[17]

An IR proximity sensor works by applying a voltage to a pair of IR light-emitting diodes (LEDs) which in turn, emit infrared light. This light propagates through the air and once it hits an object it is reflected towards the sensors.[18]

Because the hall effect sensors require the bottom of the trains to be magnetic, infrared proximity sensors were deemed to be the better candidate. A small infrared proximity sensor titled "SparkFun Line Sensor Breakout – QRE1113 (Digital)"[19] was assessed to suit all necessary requirements. And so, 28 of these sensors were purchased to be configured as displayed in figure 2-25.

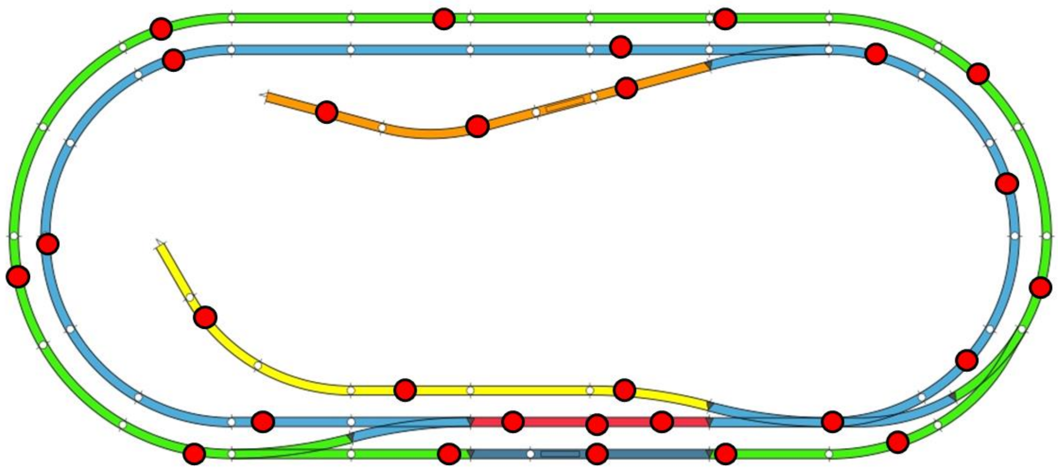


Figure 2-25

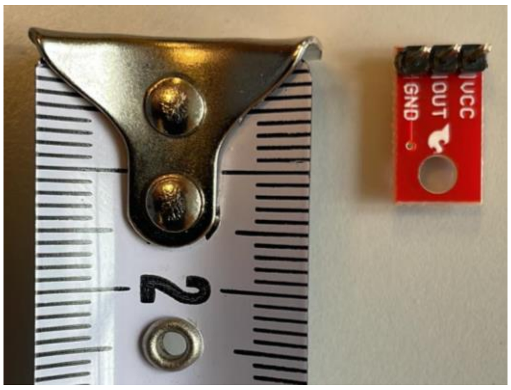
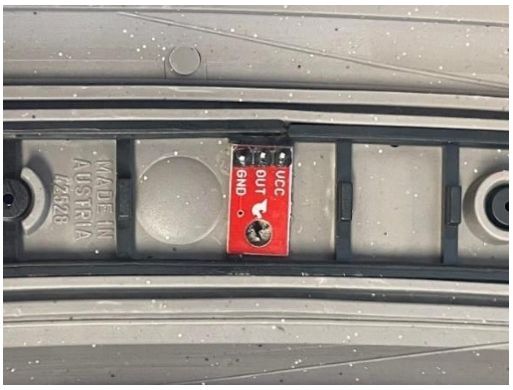


Figure 2-26 **Figure 2-27**



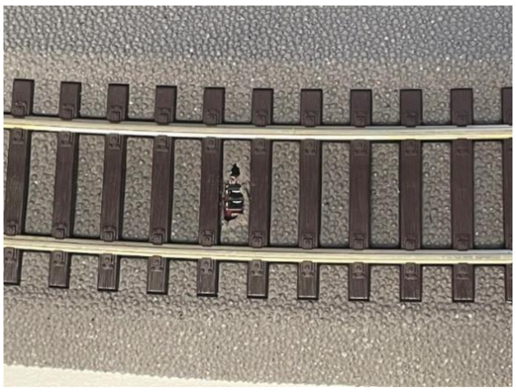


Figure 2-28 **Figure 2-29**

3.5 Wiring

All selected actuators and sensors operate on electricity. Therefore, a wiring system must be developed to facilitate the distribution of electrical energy, and the transmission of electrical signals to and from signal processing units. So, to address this challenge, two alternatives were considered.

3.5.1 Alternative 1: One Signal Processing Unit

The first alternative proposes wiring all the sensors and actuators to one signal processing unit. As displayed in figure 2-30, a simplified wiring diagram was made to establish what components must be incorporated along with the signal processing unit. This alternative does achieve the desired functionality of distributing energy and transmitting signals.

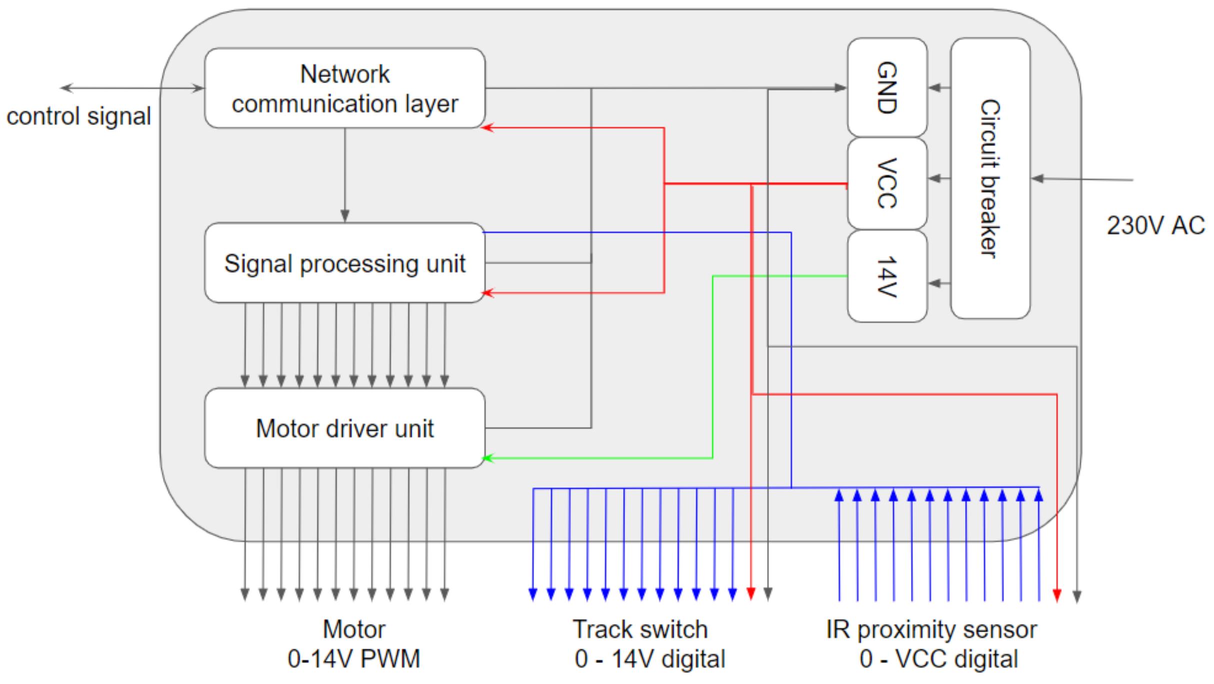


Figure 2-30

However, there are two significant limitations to the proposed solution. It requires that the selected signal processing unit has enough I/O pins to accommodate the solution and that a significant length of cabling and wiring must be used to connect the components. Another concern is that by having only one signal processing unit, there is only one point of failure. So, if for whatever reason this unit fails, the entire control system would be rendered useless.

3.5.2 Alternative 2: Multiple Signal Processing Units

To address the concerns met with the first alternative, a second alternative was developed proposing the use of multiple signal processing units instead of just one. This ensures that there is enough I/O connectivity and that there is not a single point of failure at the signal processing level. Also, by having multiple signal processing units, they can be distributed to lay closer to the actuators and sensors. This serves useful to limit the amount of cabling and wiring used.

Another advantage introduced with this alternative is the possibility of tailoring each signal processing unit to a specific task. Three different such purposes have been proposed, as displayed in figure 2-31. A decision was made to incorporate a *black box* philosophy to the signal processing layer*.*

> *A black box refers to a system whose behavior has to be observed entirely by inputs and outputs.[20]*

Within this context, that means that it should be possible to operate the control systems without having a comprehensive understanding of PCB's, motor drivers or other complicated aspects of the signal processing chain. To demonstrate the idea, consider how an operator would control the motors: *Turn X switch off* or *set Y motor to Z speed.* In other words, input values produce a desirable output signal without the operator needing to understand the control system logic.

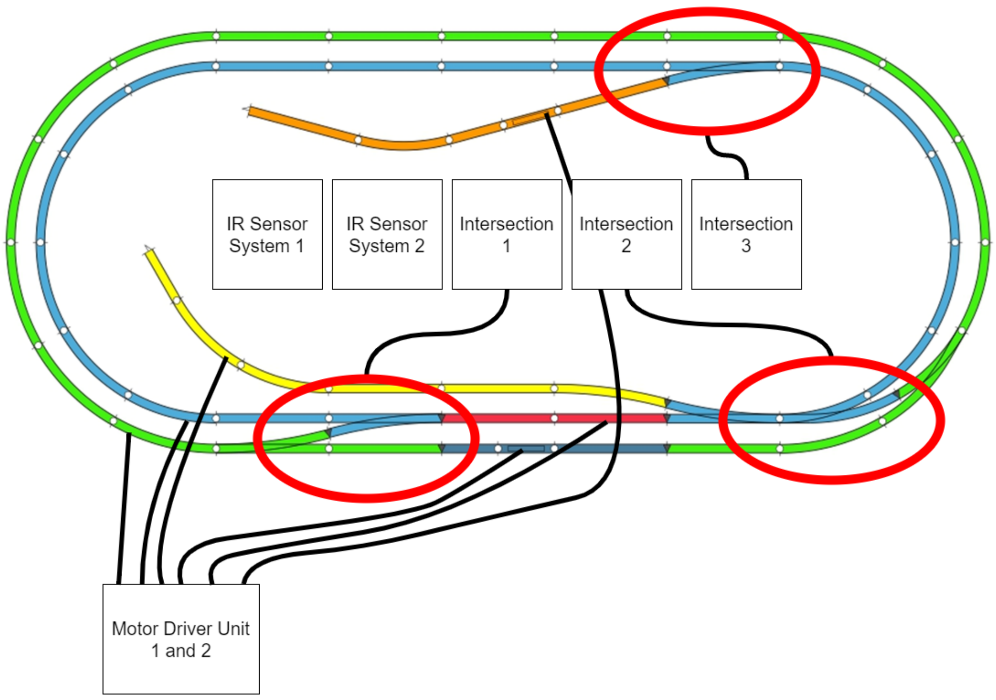


Figure 2-31

3.5.2.1 Motor Driver Unit

To control the movement of a train, the motor driver unit translates a control signal into a PWM-signal and a digital signal to control the speed and direction.

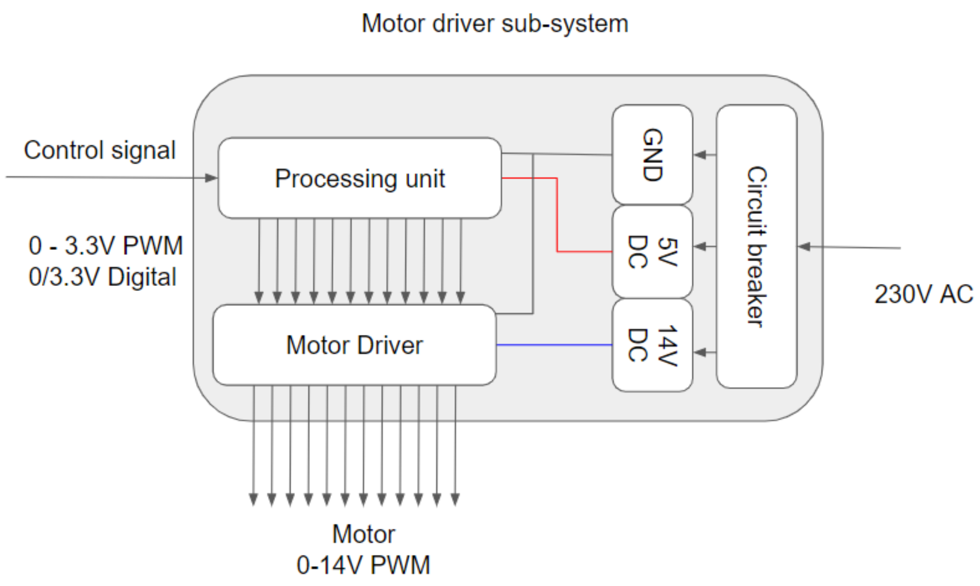


Figure 2-32

3.5.2.2 Intersection Unit

The intersection unit translates a control signal into a digital signal to control the intended track switch actuator.

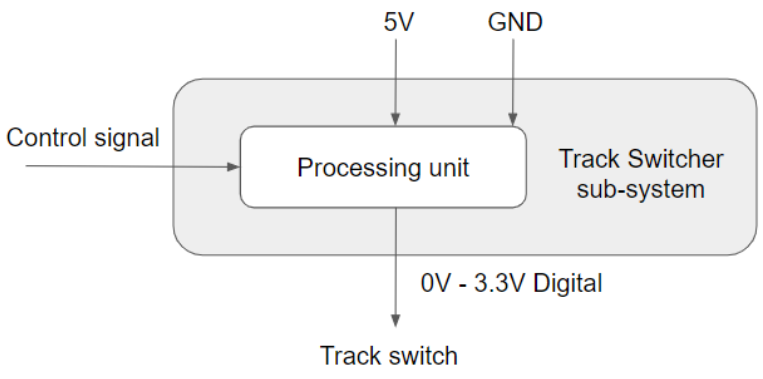


Figure 2-33

3.5.2.3 IR Sensor Unit

The IR sensor unit translates a digital signal from an IR proximity sensor into a control signal to relay the localization of a train.

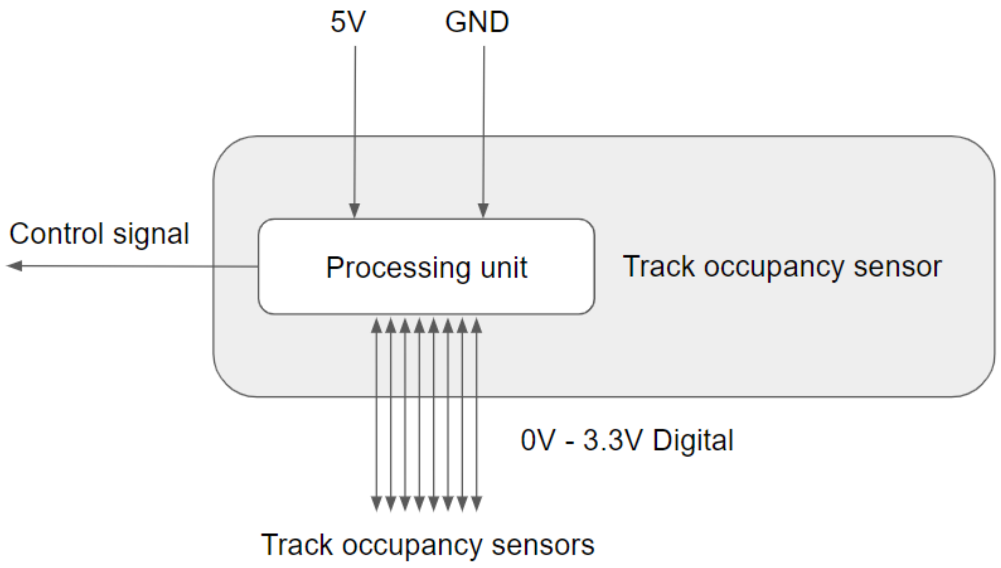


Figure 2-34

3.5.3 Electrical Control Panel

To distribute electrical energy throughout the wiring system, it was decided to utilize 230-volt alternating current from a nearby electrical outlet as a source. This presents the challenge of safely converting the alternating current to a designated direct current at specified voltage. Because electricity at 230 volts can be deadly when not handled carefully, there are strict regulations that must be adhered to. In compliance with NEK400:2022[21], it was deemed appropriate to develop an electrical control panel. And within the panel, a circuit breaker, two transformers, two motor driver units, a signal processing unit, a power switch, and some DIN-attached screw terminal blocks were installed. The list of incorporated components can be studied further under appendices.

To address the possibility that a wireless signal processing unit might be incorporated, an antenna was incorporated at the bottom of the enclosure. This was to accommodate for the possibility that a grounded metal enclosure, acting as a Faraday cage[22], could prevent the transmission of a wireless signal. Another concern was that stranded wires could be dislodged from their points of termination. Therefore, wiring ferrules were crimped to the ends to ensure proper termination in screw terminals. By implementing detachable connectors at the bottom of the enclosure, the panel can be safely disassembled from the support structure whenever necessary. GX16 connectors, commonly referred to as aviator connectors[23], were selected as they suit the criteria.

There were some slight differences between the proposed solution and the final implementation. Specifically, the order of the DIN-rail attached screw terminal blocks. Also, to ensure adequate separation between the jumper bars connecting the screw terminal blocks, separators were implemented between the differently colored ones. The color scheme of the screw terminal blocks is as follows:

Black	Negative pole, 5 and 14V DC
Red	Positive pole, 5V DC
Yellow	Positive pole, 14V DC
Brown	Phase, 230V AC
Blue	Neutral, 230V AC
Green and yellow	Protective earth

Label	Component
1	Signal processing unit
2	Motor driver unit number 1
3	Motor driver unit number 2
4	230V AC to 14V DC power supply
5	230V AC to 5V DC power supply
6	Circuit breaker – C10A

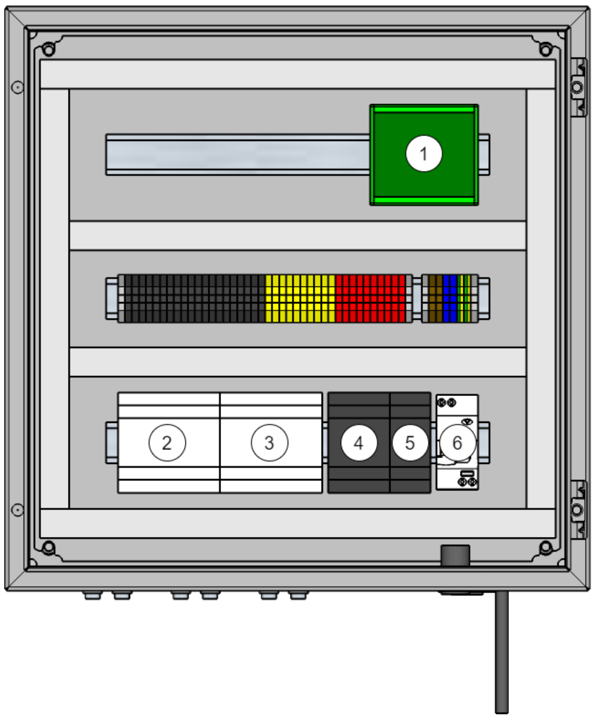


Figure 2-35

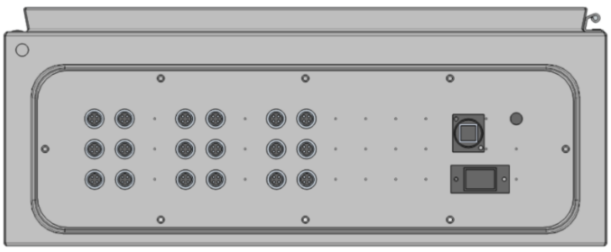


Figure 2-36

Figure 2-37

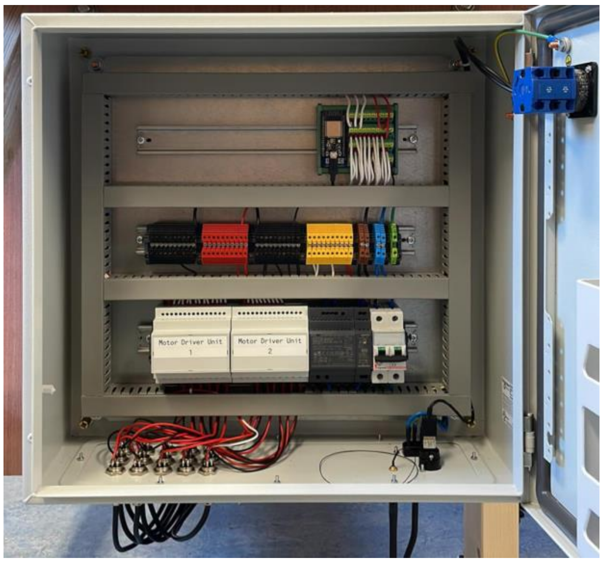


Figure 2-38

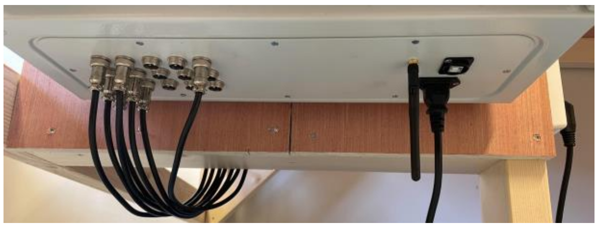


Figure 2-39

3.5.4 Junction Boxes

To address the wiring underneath the tabletop of the support structure, eight junction boxes were installed. Within these junction boxes, both signal processing units were installed, and wires terminated. Five and fourteen volts of direct current were distributed in parallel using Wago 221-connectors[24]. When necessary, an electrical drill was used to drill holes through the tabletop so that wires could be connected to the actuators and sensors above.

Figure 2-40

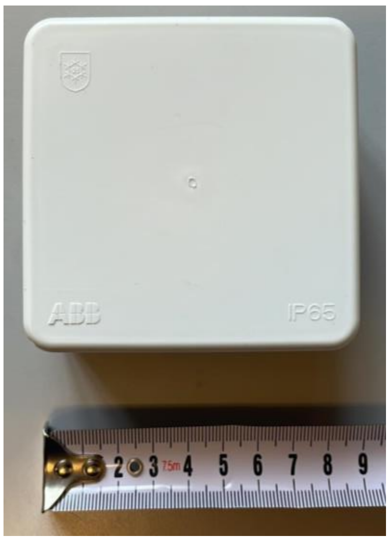


Figure 2-41

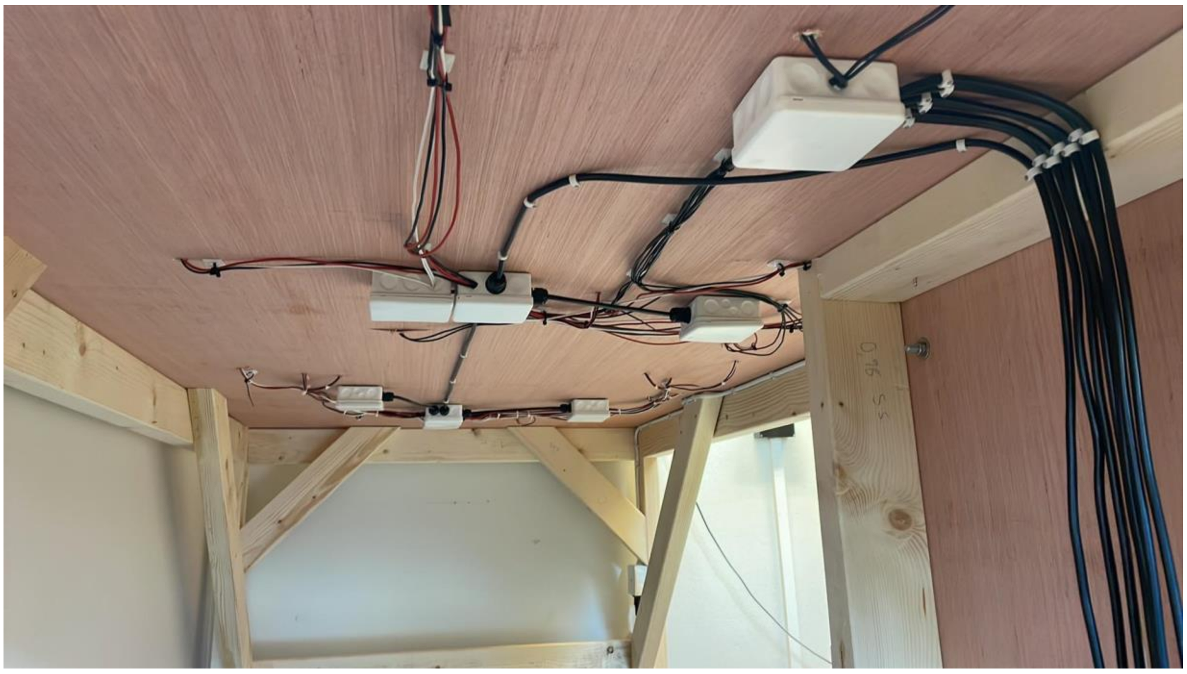


Figure 2-42

3.5.4.1 Medium of Control

Machine-to-machine communication allows each process to have access to any data from any other process in the system. Instead of viewing a processing unit as a standalone machine, it can be regarded as a component of a larger machine. That the network itself can be approached as being one process. To facilitate the communication between processing units there needs to be a medium by which they are connected, and rules the communication follows that both sides agree to. This section focuses on the transmission and reception of control signals by the processing units.

A constraint for this project was the need to control the system from a mobile device, and to not have a gateway to the internet. The wireless networks on location are a part of and managed by the local municipality with strict rules. All processes **must** interface with the same broker, this includes the web application a user might open from their mobile device. Mobile devices rely on wireless communication due to limited I/O. Consequently, at least one part of the system would have to interface with a wireless entry point. To keep the interface common between all processes and simplify the project, *all* processes will use wireless communication.

3.5.4.2 Selection of Signal Processing Units

Since this project wants to leverage wireless communications there are implications in the choice of microprocessors. Not all microcontrollers come with the hardware needed to access Wi-Fi. To do so there are two options, use an external Wi-Fi module or a microcontroller with a built-in Wi-Fi chip. Development kits often come with a USB interface in addition to the general-purpose pins. But for the microcontrollers to be able to interface through for example an ethernet cable, peripheral modules based on the W5500 ethernet controller would be required.

Microprocessor Unit	**Arduino Uno**[25]	**ESP8266**[26]	**ESP32**[27]
Flash	32 kb	2 Mb	4 Mb
CPU Clock	15 MHz	80 MHz / 160 MHz	160 MHz / 240 MHz
CPU Cores	1	1	2
2.4 GHz Wi-Fi	No	Yes	Yes
Bluetooth	No	No	Yes
Price	24 €	5.15 €	8.50 €

Looking at the specifications, some of which are listed in the table above, are other factors considered in choosing microcontrollers. Since the ESP family is cheaper than official Arduino boards. It meant getting more hardware for the same amount of money. This is without the additional costs associated with the Wi-Fi module Arduino boards would need.

Most ESP32 microcontrollers do not yet have support for 5GHz Wi-Fi frequency, only 2.4GHz. Though a model with support for it was released in 2022, it is not yet widespread. The consequence of this is that all the microcontrollers must share in the same lower capacity band of the router.

For these reasons, a combination of the ESP boards is used in this project. The ESP boards have output voltage limited to 3.3V as was highlighted in the discussion on the selection of motor drivers.

3.5.5 Final Implementation

Wireless is fantastic for the purpose of development by reducing the complexity of wiring and providing a universal interface. When all subsystems can connect to a common network it makes integration and testing different interactions much easier. The faster development process, the flexibility and the ease of use provided by wireless networks is why all parts of the system utilize it.

A “TP-Link Archer AX1500 Wi-Fi 6” Router was acquired for this project. Modern routers like the AX1500 are fast, Wi-Fi 6 can deliver gigabit per second speeds.[28] Specifically, through the 5GHz connection on the Archer router where the base speed is 1.2GB/s.[28] The router advertises a 300Mb/S speed on its 2.4Ghz connections.[28]

3.6 Cooperating Processing Units

3.6.1 Internet of Things

The Fourth Industrial Revolution, *Industry 4.0*, encompasses a range of advancements in the fields of embedded system technologies, automation, and data exchange to drive efficiency, productivity, and responsiveness. One key aspect of Industry 4.0 that holds significant relevance to this project is the concept of the Internet of Things (IoT). IoT entails the interconnectedness of devices, sensors, and systems, enabling seamless communication and data exchange. This aspect of IoT is known as machine-to-machine communication and is what enables the processes in this system to cooperate with each other. By leveraging IoT technologies, real-time monitoring, analysis, and control of the system can be achieved.

3.6.2 Communication

The movement of data throughout this system is an important part of this project. There are many methods and techniques that can accomplish this. To ensure that communications reach their destination it is desirable to leverage the TCP communications standard. The TCP standard includes mechanisms for error detection via check sums and data handshaking, which ensures the integrity of the data when it reaches its destination.[29]

TCP achieves transportation of data by breaking it down into smaller packet which can be reliably transmitted.[29] TCP does not care about the contents of the data only its representation in bytes. To implement the complex behavior planned for this project, it is up to the developer to implement systems for parsing and sending this raw data. On top of TCP, suitable *communication protocol* is desirable to achieve effective machine-to-machine communication.

3.6.2.1 Communication Protocols

Communication protocols let clients agree on the rules with which they communicate. These rules are defined in standards to make sure all clients behave the same. Using standardized protocols and interfaces ensures interoperability and compatibility between diverse devices, systems, and platforms. Protocols, such as OPC UA (Unified Architecture), MQTT (Message Queuing Telemetry Transport), and CoAP (Constrained Application Protocol) were considered. These protocols come with their own strengths and weaknesses. By adhering to these standardized protocols, processes can communicate and share data reliably and securely, regardless of their underlying technologies or manufacturers. Having chosen wireless as the medium of communication wired bus-based protocols like EtherCAT or PROFINET are out of the conversation.

To make the right choice for this use case there are things to keep in mind. This system will consist of multiple units who want to enact changes in the system, these are the users. There will also be microcontrollers that want to communicate changes in their states and respond to other clients' actions. In other words, there are many processes that want to communicate with many other processes. For example, if there are multiple users and one of them decides to accelerate a train. This information is wanted by other users, the responsible microcontroller, the database, and the control system. The following chapters give a more in-depth discussion on whether to use HTTP and MQTT.

3.6.2.2 HTTP

"HTTP (Hypertext Transfer Protocol) is the underlying protocol of the World Wide Web" [30]

This quote from the Mozilla foundation is reason enough to consider HTTP as the basis for this project. Websites are traditionally written with HTML (Hypertext Markup Language) precisely what HTTP was designed to transfer over the internet. Servers hosting HTTP content are referred to as *Webservers,* and the HTTP model is request and response.[31] Given the address to a to a webserver a client can make a HTTP request, and then be served a hypertext document or JSON object that is rendered in the browser. If a change is made to the data used on the webserver, the default behavior is that this will not be sent to a client unless a new request is made. Because the client has not yet requested the update. The consequence of this behavior is that all clients need to continuously make requests to ask if any changes has occurred, a method known as *polling*.[32]

The HTTP standard defines *methods,* a way of modifying the requests.[33] These enable more ways of interactions between server and client. Like altering or deleting data. The HTTP protocol has changed throughout the years. Earlier versions of HTTP utilized TCP for transportation but since version 3 it utilizes a newer but less familiar network protocol called QUIC.[30] For which the tooling is less widespread. This would mean versions prior to 3 would be the ones considered for this project.

The desired *many-to-many* model desired would be difficult to implement with HTTP, though there are ways. For an HTTP request to alter the state of a microcontroller there is a challenge. One way is to create a webserver to which all processes connect to and interact with through polling, this is possible although difficult. The routing of data and interactions between processes would have to be implemented for this project. Another way would be for all the microcontrollers to host webservers of their own that users communicate with directly through web applications. For example, to update the speed of a motor, a user might send a *put* request with its new value to the webserver hosted on the associated controller. But this might be a lot of overhead for the weak microcontrollers. To actively communicate changes in state to all interested clients they would all need some level of polling to the controller too.

While it is possible to utilize HTTP for this system via polling methods, it would require a lot of work implementing message routing and queuing. The request-response model HTTP is designed around, having one server and one client, is inherently antithetical to the desire to disseminate data throughout the system. This is not to say that HTTP is not of use in the project, it will be used to deliver the UI to the end user. But the real-time communications aspect of the project will not rely on it. As will be discussed in chapter 2.7.3, the history of the system will be accessible through a RESTful API through HTTP requests for observability purposes.

3.6.2.3 MQTT

Researching IoT projects, *MQTT* seemed popular. MQTT advertises itself as "an extremely lightweight" communication protocol and is designed based on a *Publish/Subscribe* model.[34][35] All processes that want to participate with MQTT are referred to as *clients*.[35] Clients declare what information they are interested in by *subscribing* to a *topic*.[35] Clients that produce information, may *publish* that data to the relevant topic.[35] Multiple clients may publish or subscribe to the same topic, facilitating the dissemination of data throughout the system.

Consider the example of a client sending data to control a motor. The parties interested in this information include the database who is logging the activity on the network, the relevant motor controller, other client user interfaces, the system controller that prevents collision. To receive updates the clients need only subscribe to the topic where changes in speed is communicated.

MQTT offers a lot of support and functionality not often seen in other communication protocols. Message queuing, message routing, message priority, client authentication, known connection state, known last good value, and more.[31] The reason MQTT can offer these features is because to use it, all clients need to be connected to the same *broker*. Another benefit as compared to HTTP is the bandwidth required to send the same message. HTTP requests contain a lot of information about the sender in the header of the request, while MQTT includes little. This means size of a HTTP request is much larger than the same message in MQTT.[31] For its ease of implementation and powerful features MQTT seems good for this project.

3.6.2.3.1 MQTT Broker

The MQTT protocol requires a broker to function.[35] The broker is a program that implements the functionality specified in the MQTT OASIS standard. All programs that want to communicate must interface with the same broker. The broker keeps track of which of its connected clients have subscribed to what topic.[35] When a client publishes a message to a topic, this message is sent directly to the broker who in turn is responsible for routing it to the relevant clients.

Like in TPC handshaking is an important aspect of MQTT, all requests to the broker are *acknowledged.*[35] When a client publishes data, the broker will respond to the client to the responsible client. When routing a message to a client the client also must respond if the data has been received. This way all clients can know if their messages were received as intended, or if something abnormal is happening to the network.[35] These acknowledgement packets are not the only behind the scenes action performed by the broker. It also ping's all connections at frequent intervals to determine whether the clients are alive.[35] These benefits are gained by the system for free by just using MQTT.

Developing a standard compliant broker is outside the scope of this project. Luckily, there are open-source projects or commercial products available. For this project, no functionality beyond what is specified in the MQTT standard needed. Meaning that whatever broker would work fine so long they are up to specification. To avoid licenses or any legal issues, an open-source alternative would be preferable.

This project will only accommodate a dozen or so clients who themselves will not be sending a large amount of data. Mosquitto is an open-source implementation of an MQTT broker along with simple command line clients.[36] It is a project of the Eclipse foundation[37] who is an independent non-profit corporation. Mosquitto is easy to use with implementations for both Windows and Unix environments. It can be run as a binary on windows or as a command line program in Unix systems needing little configuration. The contents of the configuration file can be found in the appendix. But the most important configuration is letting anonymous clients connect to the broker, and opening a port designated for communication through Web Sockets. Both configurations are needed to allow clients to connect from a web browser.

Using Mosquitto made development for MQTT straightforward. Starting a command line instance of the broker, then providing client programs with the IP address of the machine running Mosquitto always worked seamlessly. Another benefit of using Mosquitto is congruity between the vendor of the broker and clients. The plan is that the clients would utilize the Paho library to implement their MQTT functionality. Paho is another project under the Eclipse foundation.[38] While not critical, a choice was made to make sure the clients and broker were of the same origin when possible. The Paho project has created implementations for many programming languages.[39] JavaScript Paho was embedded in the web application, and Python Paho in other processes was utilized in the project.

3.6.2.4 Messages

Having figured out how programs in the system interface with one another it is equally important to decide what the data sent will contain. Continuing with the motor controller as an example, its software was designed to only need the value of the desired speed, and which motor it affects. But the motor controller is not the only party interested in this data, the database is too. The database, however, also wants to know who sent the data. To satisfy all clients it needs to contain three things. The target motor, the target speed, and the unique id of the sender.

The semantics or syntax that represents this message is also relevant as it decides what methods the clients need to use to extract relevant information. There are many ways to represent this speed command message, popular plain text representations of objects include XML, JSON, and YAML. With JavaScript's prevalence in the world of web development, it was chosen. Having interacted with web APIs through prior education, the experience is that data most often is JSON format. Experiences like this might have led to a belief that data *should* be represented with JSON which solidified the choice. Represented in the JSON format, the command to change speed looks like this:

```
{"Id":0,"Speed":0,"Sender":"8871e568-e56a-4a82-9acb-40cddee349f2"}
```

Where *Id* refers to the motor whose speed is to be changed. *Speed* is the target value of the motor in the range -100 to 100. And *Sender* is the unique identifier of the client from whom the command was sent. Clients need only access the key value pair in which they are interested. Examples of other commands can be found in the appendix.

3.7 Software and Interfaces

In this project many processes come together to form a system greater than the sum of its parts. The following chapters describe the main functionalities of and gives an overview of how and why each process was implemented. The processes contribute toward the goal of implementing industry 4.0 concepts and digitizing the railway.

While security is an important aspect of industry 4.0, the client has specified that this implementation should not focus on it. This means that no user authentication should be needed to access the model railway UI since this would get in the way of the user experience. It also means that no validation is required to interact with the broker even though the broker supports username and password control.

3.7.1 User Interface

To control the model railway the user needs some sort of interface. Model railways are traditionally analog systems and the interface for these systems is often a set of physical dials and switches. For a digital system, a digital interface is natural. Boiling it down, the functionality required from the UI is very straightforward. It needs to have a set of dials and switches, and when they are used, send a little data to the responsible microcontroller with MQTT.

3.7.1.1 Alternatives

To create an interface that would be accessible from a mobile device, it would have to be an app or a website. Developing an app comes with considerable overhead. There are frameworks for developing apps that work on both android and IOS so the development might go fine. But to make an app accessible on android and IOS it would need to be submitted to the respective marketplaces for approval.

Since the UI only needs to be available while on the location of the railway, it can just be hosted on a machine on the local network. Choosing between developing an application or website, a website seemed the easier and more versatile choice since a browser is all that is needed to access it.

3.7.1.2 Web-Development

How one goes about creating and monitoring the dials and switches on the page that controls the system is not important. Any framework or structure could achieve that. Because of lack in web development experience, an approach that was said to be on the easier side was chosen. [40] The *SvelteKit* framework stood out. With it you write *svelte* files. In your svelte files you write locally scoped JavaScript, HTML and CSS in the same file. Then you use a compiler to convert them to .JS, .HTML and .CSS files. It was very intuitive and made for productive development.

One crucial tool for the functionality behind the UI is the JavaScript Paho library. This library implements the MQTT standard, making it trivial to subscribe and publish to topics from the website. Given the address to a MQTT broker, the *Paho* library opens and manages a web socket connection between the browser to the broker.

3.7.1.3 Design

Figure 2-43 showcases how the control panel on a sufficiently advanced model railway system might look. The lower part of the picture demonstrates how switches embedded into the representation of the track clearly indicate what switch is responsible for which intersection. The interface designed for this project is intended to emulate these kinds of control panels.

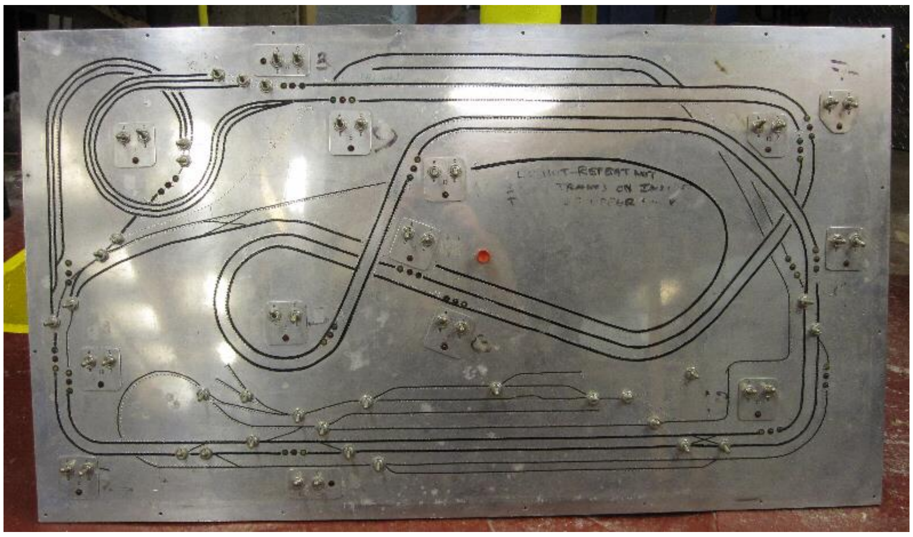


Figure 2-43

Figure 2-44 is an early mockup of the website that would allow for the control of many zones and many track switches. Speed and direction being controlled by the slider at the left, accompanied by a stop button. The color of the active tab is used to illustrate what zone is being affected by the users' changes. On the right side there was a map showing the layout of the track accompanied by buttons that indicate in which of the two possible states the switch is currently

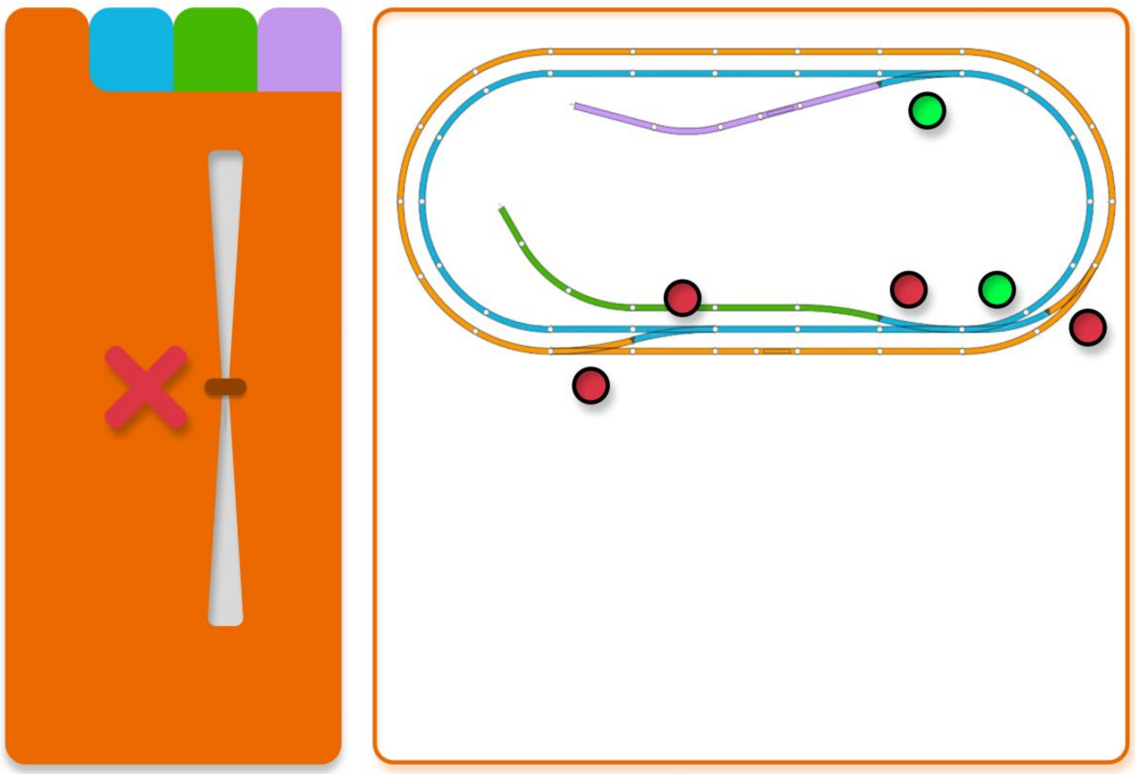


Figure 2-44

The user experience of mobile devices was not great in the proposed configuration, since interacting with the right and left most parts of the UI in quick succession was difficult. To improve this only the speed controller or switch controller was shown at a time, and a mechanism for toggling between them was added.

Figure 2-45 and 2-46 show the implementation of the current design on the final UI web page. There were additional differences between the early design and final implementation. The styling on the speed slider seemed to vary based on the browser the page was viewed in, and trying to make it uniform across them was more challenging than expected.

Having the color represent what zone was currently being initially seemed like a clever way of doing it. But when the color of the zone was like the stop button or the slider it did not look great and potentially posed accessibility issues. Functionality for displaying that a sensor has detected *something* was also added, when a detection is reported a little sensor emoji appears on the map where the sensor is located as illustrated by figure 2-47.

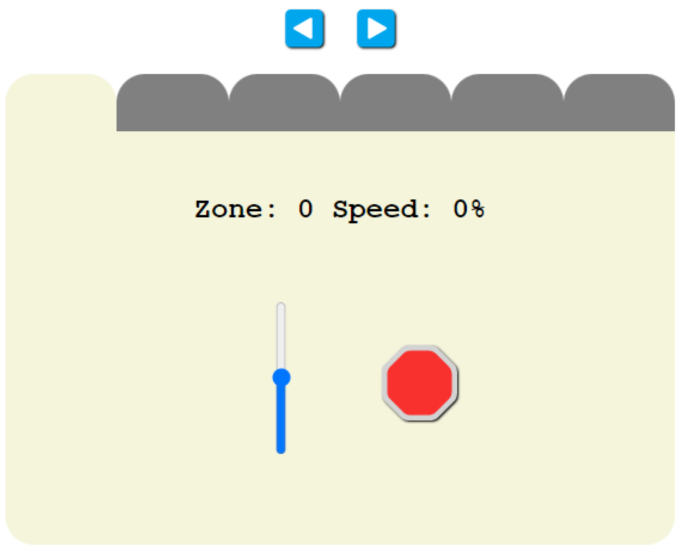
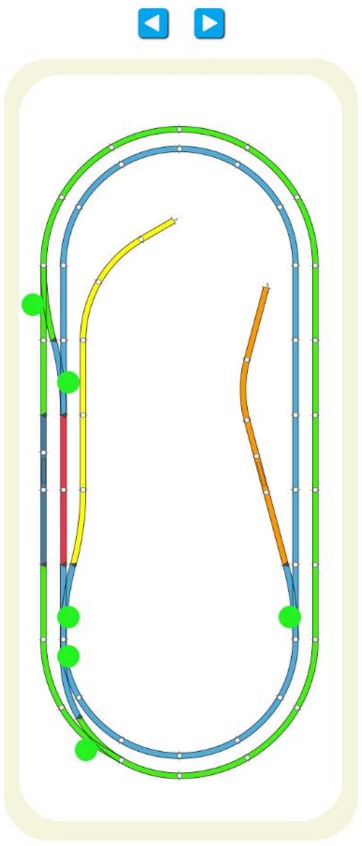


Figure 2-45



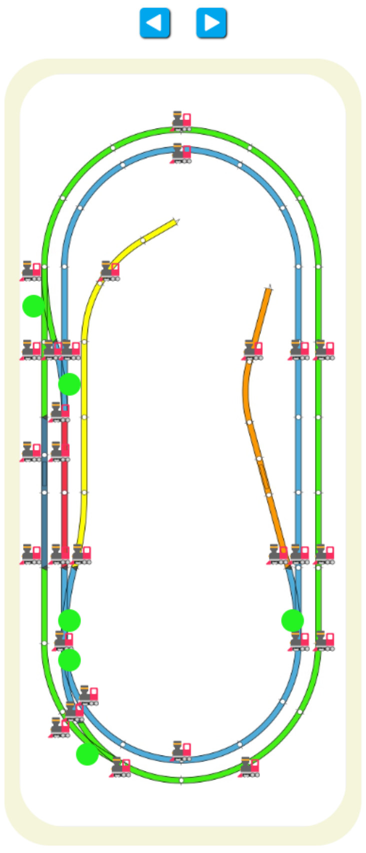


Figure 2-46 **Figure 2-47**

3.7.2 Database

If something goes wrong causing the system to break down, it is desirable to know how it went wrong and what caused it to go wrong to rectify it. This *data* around downtime and what might have caused it, is needed to maximize uptime. This data driven approach to systems is central to industry 4.0.[41] Since data is ethereal, then deliberate action must be taken to hold on to it. A database management system provides the tools to access and save data effectively.

Tracking what messages have been sent and saving them to a database. The specific database system is not important, SQLite was chosen for this project mainly because of prior experience with it and there is a very user-friendly python library for it. The database schema can be viewed in the appendix.

3.7.2.1 MQTT Bridge

With a database, a mechanism to populate it with data is also needed. The role of the MQTT bridge is to observe and save all activity going through the MQTT broker. To achieve this, it needs to do three things. It must connect to the MQTT broker, connect to the database and parse MQTT messages into a format appropriate for the database.

This program uses the python Paho library to connect to the broker and subscribes to all the topics. All data is sent in JSON format and is easily parsed with the python JSON library. SQLite3 also has a python library, making it easy to interact with an SQLite database. The program is meant to work one way, sending data out of the MQTT system.

3.7.3 Application Programming Interface

Since there is a database, a way of interfacing with it is desirable. To make it easily accessible, a REST API with the FastAPI python library has been implemented. FastAPI streamlines the process of creating API endpoints and what happens when a request hits a given endpoint. This API has direct access to the local database through the SQLite3 library. This allows access to the data on the database through this API using HTTP methods.

If a new user opens the UI while another is interacting with it, then the interface given to the new user will not represent the current state of the system. It will correctly update any new events, but the application has no knowledge of what has happened on the system before. The API is intended to also solve this problem. When a user first loads the web page, it automatically queries the API for the up-to-date state of the system.

3.7.4 Microcontroller Unit

Microcontrollers are the mechanism that allows this project to bridge the system from analog to digital. The software built for all the microcontrollers follows the same basic structure. Continuously checking whether any data has arrived. When data arrives, it is then parsed and written to memory. Another section of the program continuously looks at this memory, and based on the value therein makes changes to the system I/O.

The microcontrollers in this project does not use the Paho library, but rather the PubSubClient[42] library which is another implementation of the MQTT client. This is for the simple reason that the number one priority when the system was developed, was to ensure that the microcontrollers could effectively use MQTT. When investigating how to use MQTT with microcontrollers this library was the most well-documented. Since it worked well there did not seem to be any reason to migrate to another solution.

3.7.4.1 Motor Driver

Figure 2-48 represents the code written for the motor controller. A *Motor* class was created in C++ that, amongst others, contains the properties *Target* and *Current* which are the most frequently used properties. The same class also includes a *Run* method. This method evaluates the difference between *Target* and *Current*, then changes the *Current* value to be some amount closer to *Target*. The Run Method is called many times a second and will over time make the values equal. This is implemented to make the trains accelerate in a more life-like manner. The motor object is also instantiated with the output pin information associated with the motor it represents, meaning that when computing the Run method, it also manages writing the *Current* value to the correct pin.

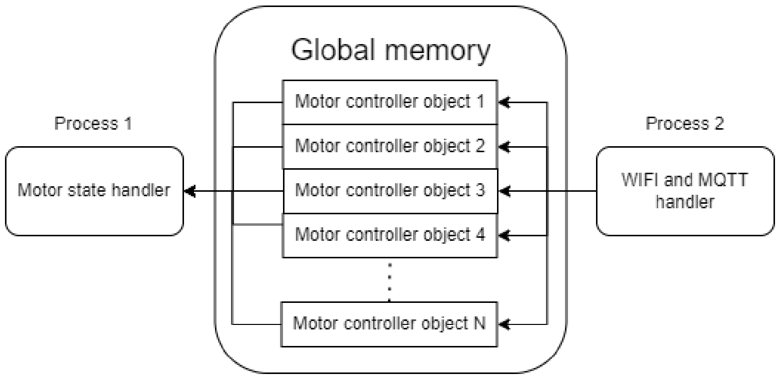

Global memory
Process 1
Motor state handler
Motor controller object 1
Motor controller object 2
Motor controller object 3
Motor controller object 4
Motor controller object N
Process 2
WIFI and MQTT handler

Figure 2-48

There is a nuance specifically with the motor driver that is not in the other controllers. When adjusting the speed of the train, the value *Target* is modified, not the output. Which the value Current, eventually will become. If the microcontroller loses connection with the MQTT broker, meaning that the control of the train is lost, then the train should stop on its own. But in the same way where it is adjusted over time. Since in this situation the connection with the broker is lost, the microcontroller should attempt to reconnect. Attempting to reconnect can either succeed, or the request will time out. While waiting for the request to complete or time out, if the motor *Run* method is processed on the same thread, then each iteration of the *Run* method must wait for the reconnect request to time out. Meaning that instead of the *Run* method being processed many times a second, it might only be processed once every one or two seconds. This dramatically increases the time before *Target* and *Current* become equal. To solve this problem, another Arduino Loop function whose only job is to evaluate motor state is needed. The ESP SDK is built on top of the Free Real Time Operating system (FreeRTOS).[43] Part of that operating system is a scheduler that distributes processing time between running programs. FreeRTOS provides an API for the creation of *Tasks* whose functionality is just like the Arduino loop function.[44]

3.7.5 Control Algorithm

The control law is responsible for achieving the desired *behavior* of the railway system, truly achieving cooperation between processes. The program should contain a digital representation of the railway and listen to all the messages sent on the system to maintain an accurate representation. For example, if it receives sensor readings that indicates two trains are entering the same intersection. Then it will send a command to one of the trains, making it slow down or stop. There are two main components to this program which was implemented with python. Connecting to the MQTT broker and managing a complex state machine. To connect to the MQTT broker it uses the same method as the MQTT bridge.

3.7.6 Process Overview

This project is comprised of many processes that interface with one another. Figure 2-49 gives an overview on how all the processes in the system interface each other.

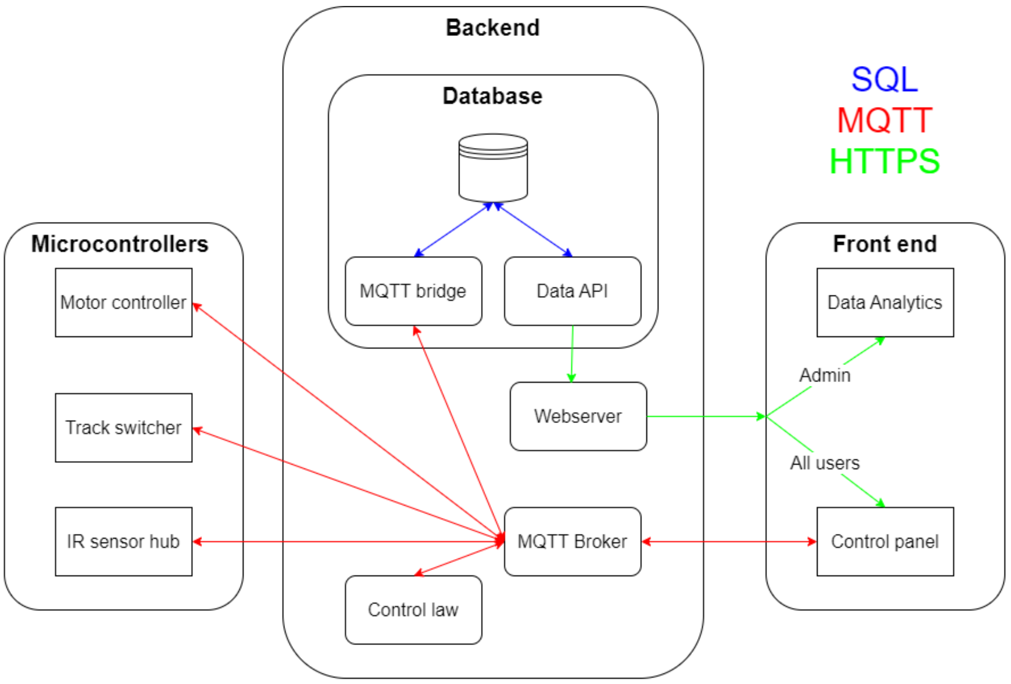


Figure 2-49

4 Results

4.1 Wiring

A system for distributing power to actuators, sensors, and microprocessors was successfully implemented.

4.2 Motor Driver and PCB

The circuit for driving the trains worked as intended. This has been tested by placing a train on the track and sending a command to power that section.

4.3 Track Switching Actuator and Traffic Lights.

Designing a circuit needed by the microcontroller to control the actuators proved difficult and was not achieved. This in turn meant that the processing unit associated with the actuators could not be correctly implemented, and neither could the traffic lights.

4.4 Track Occupancy Sensors

The sensors were embedded in the track of the system and successfully respond to passing trains.

4.5 Web Application

The web application was successfully created and has the capability to send and receive data through MQTT. Getting the most up to date system state when connecting by querying the API was not achieved. No UI was developed for data visualization.

4.6 MQTT Bridge

The MQTT Bridge can successfully connect to the MQTT broker and saves network activity to the SQLite3 database, achieving all its goals.

4.7 API

The API was successfully created and can be queried for data. Though there are few endpoints. The active endpoints can be viewed in the appendix.

4.8 Processing Units

The processing units responsible for controlling the train motors and track occupancy were implemented successfully. The software for the track switcher unit has been written in line with the other units, but due to the issues encountered with the actuator it has not been fully implemented.

4.9 Control Algorithm

The control algorithm was not completed. Though the program could correctly communicate with MQTT, development and testing were not prioritized since the track switching actuators were not implemented.

4.10 Requirements Fulfilled

This section is a reflection on whether the goals of the project were achieved. Goals are presented as bullet-points while reflections follow.

- The control system should allow users to operate track switches, traffic lights and control the speed and direction of model trains on the multi-train model railway.

The user can only control the speed and direction of the model trains.

- The control system architecture must incorporate a carefully selected set of actuators, sensors, and other relevant components to meet the specific requirements of the existing model railway set.

The sensors and actuator were chosen to effectively meet the requirements of the project.

- The system must embrace the principles of Industry 4.0, integrating cooperative processing units that enable seamless collaboration between various components.

The system achieves cooperation between the processing units with M2M communication through MQTT.

- Digitalization of Existing Model Railway Set.

Certain aspects of the model railway have been digitized such as the train direction and speed. Though track switching has not been achieved.

- Wireless Control and Communication.

The infrastructure and semantics of wireless control have been implemented with MQTT.

5 Discussion

The results presented in the previous section have implications for the project. This discussion section will expand on what they are. It will also highlight the limitations of the original design, and how these might be overcome.

5.1 Model Railway Control:

The system that has been developed in this project for controlling the speed and direction of the trains worked well. A user may interact with the railway system through their mobile device as planned. This is achieved with IoT techniques like M2M communication with the MQTT technology.

The users’ control of the system is limited due to the zone-based control of the trains. Direct control of the trains is an alternate strategy that could be interesting to pursue. This would provide the end user with finer control of the system, improving the user experience. It would for example allow two trains in the same zone to have different speeds. The internal workings of the trains would need to be changed to apply this strategy. The main modifications would entail decoupling the internal motor from the wheels, where they currently get their power, and providing some kind of on-board control mechanism.

Another thing limiting the users’ control of the model railway is the difficulties encountered with the track switching actuators. Getting the microcontroller to actuate the switch was more difficult than anticipated. Since the actuators arrived late in the project, there was not much time left to design a circuit that would allow the microcontrollers 3.3V outputs to drive the two 14V signal pins of the actuator. Future work could either investigate how to solve that problem or design a more customized solution using another kind of actuator.

5.2 User Interface

The UI allows for the user to send signals to control the motor and track switcher actuators, in addition to view when and where IR proximity sensors activate. A limitation of the UI is that when a user opens the website, it might not reflect the state of the track if another user has been using the system. This can be improved by making the application reach out to the API and request the information when first loading.

5.3 Data

All the mechanisms to save and retrieve data about the system have been built and behave as expected. This includes the MQTT bridge, the database, and the API. While the data is available, no tools have been developed to visualize or analyze it. Statistics about a session would improve the user experience. It might also be interesting in the context of an Industry 4.0 system where analysis of system behavior is important.[41] In relation to data analysis, the API could be improved by developing more endpoints to better facilitate advanced queries. One endpoint could for example be created to return the entire system state in one go.

5.4 Control Algorithm

No control algorithm was implemented. This process would be responsible for stopping trains from colliding or automatically regulating their speed in response to sensor data. The consequence of this is that there are no guardrails on the system to prevent collisions.

Since the track switching actuators were not implemented, the system had no way of knowing what the state of the intersections were. Without the intersection state being known it would be impossible to determine whether two trains would collide. If the state could be known without the need of the actuator there would be a way to have implement the control algorithm effectively. More complex behavior could also be implemented in the control algorithm such as fully autonomous control of all the trains.

5.5 Track Occupancy Sensors

The IR proximity sensors work well, in the sense that they successfully detect trains as they pass. However, the efficacy of these sensors as it relates to preventing collisions, which was their original purpose, is yet to be determined. The control algorithm was intended to use the data from these sensors to prevent accidents. But since the control algorithm was not implemented, then no statement on the effectiveness of these sensors in that context can be made.

If the IR proximity sensors prove insufficient, then other methods might be pursued. One that seems interesting is computer-vision based tracking. Integrating object detection and machine learning would go well with the projects theme of industry 4.0. Images of the model railway and the positions of the trains could be clearly captured by positioning a camera above the model railway. This could yield more accurate and powerful detection of the trains.

6 Conclusion

This thesis details how analog systems such as a model railway can digitized through industry 4.0 technologies. Most of the goals set for this project were met, though problems were encountered that prevented the completion of them all. The finished system incorporates sensors and actuators and runs on a multi-zoned track configuration. Mobile devices and other digital processes can control the direction and speed of the trains through wireless communication.

This project has been a valuable experience as it provided an opportunity to learn new skills and hone old ones. Web development, microprocessor programing, M2M communication, electronics and woodworking have all played a crucial role for the project. The importance of time management and detailed planning truly became clear after this project.

7 References

[1] “Fagskulen Vestland.” https://www.fagskulen.no/ (accessed Feb. 02, 2023).
[2] “Skinnesett D, RocoLine m/ Ballast.” https://www.nmj.no/roco-42012-1010800.html (accessed May 22, 2023).
[3] “Roco company-philosophy.” https://www.roco.cc/ren/company-philosophy (accessed May 22, 2023).
[4] “What is an actuator? | REAC.” https://www.reac-group.com/en_en/facts/actuators/what-is-an-actuator/ (accessed May 19, 2023).
[5] “Model Railway Electrics - Isolating Circult.” https://www.newrailwaymodellers.co.uk/electrics-isolating-circuit.htm (accessed May 21, 2023).
[6] “L298N Motor Driver Module,” *Components101*. https://components101.com/modules/l293n-motor-driver-module (accessed May 19, 2023).
[7] “Pololu - TB6612FNG Dual Motor Driver Carrier.” https://www.pololu.com/product/713/specs (accessed May 19, 2023).
[8] “TB6612FNG Motor Driver: Better Than the L298N?,” *Hackster.io*. https://www.hackster.io/news/tb6612fng-motor-driver-better-than-the-l298n-7499a7574e63 (accessed May 19, 2023).
[9] “User manual Roco 42620 (English - 16 pages).” https://www.manua.ls/roco/42620/manual?p=12 (accessed May 20, 2023).
[10] “Types of Color Blindness | National Eye Institute.” https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-blindness (accessed May 21, 2023).
[11] M. C. U. Feb. 02 and 2023, “This Is Why Traffic Lights Are Red, Yellow and Green,” *Reader’s Digest*, Nov. 23, 2022. https://www.rd.com/article/traffic-lights/ (accessed May 21, 2023).
[12] “What Are Sensors and How Do They Work?,” *WhatIs.com*. https://www.techtarget.com/whatis/definition/sensor (accessed May 21, 2023).
[13] W. Storr, “Open-loop System and Open-loop Control Systems,” *Basic Electronics Tutorials*, Aug. 30, 2013. https://www.electronics-tutorials.ws/systems/open-loop-system.html (accessed May 21, 2023).
[14] Controlsystemintro, *English: Open loop control systems use a controller to influence the process depending on the desired output.* 2018. Accessed: May 21, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Control_systems_open_loop.jpg
[15] W. Storr, “Closed-loop System and Closed-loop Control Systems,” *Basic Electronics Tutorials*, Aug. 30, 2013. https://www.electronics-tutorials.ws/systems/closed-loop-system.html (accessed May 21, 2023).
[16] Controlsystemintro, *English: Closed loop feedback systems include a measurement entity as well as a comparator, which compares the signal from the measurement device to the desired output and sends a corresponding downstream signal to the controller in order to match measurement and desired output.* 2018. Accessed: May 21, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Closed_loop_feedback_systems.jpg
[17] “The resolution, accuracy and precision of sensors,” *Sensor Partners*. https://sensorpartners.com/en/kennisbank/sensors-resolution-precision-and-accuracy/ (accessed May 21, 2023).
[18] W. Storr, “Hall Effect Sensor and How Magnets Make It Works,” *Basic Electronics Tutorials*, Aug. 13, 2013. https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html (accessed May 21, 2023).

[19] "IR Sensor Working Principal & Applications," *Robocraze*, Jun. 20, 2022. https://robocraze.com/blogs/post/ir-sensor-working (accessed May 21, 2023).
[20] "SparkFun Line Sensor Breakout - QRE1113 (Digital) - ROB-09454 - SparkFun Electronics." https://www.sparkfun.com/products/9454 (accessed May 21, 2023).
[21] "What is a black box (black box testing)? Definition from SearchSoftwareQuality," *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/black-box (accessed May 22, 2023).
[22] "NEK 400:2022," *Norsk Elektroteknisk Komite (NEK)*. https://www.nek.no/produkter/nek-400/ (accessed May 22, 2023).
[23] J. O. published, "What is a Faraday cage?," *livescience.com*, Dec. 03, 2021. https://www.livescience.com/what-is-a-faraday-cage (accessed May 22, 2023).
[24] "GX16 Male And Female Connector – BULK-MAN 3D." https://bulkman3d.com/product/gx16-male-and-female-connector/ (accessed May 22, 2023).
[25] "WAGO 221 | WAGO," *WAGO Global*. https://www.wago.com/global/electrical-interconnections/discover-installation-terminal-blocks-and-connectors/221 (accessed May 22, 2023).
[26] "Arduino Uno Rev3," *Arduino Official Store*. https://store.arduino.cc/products/arduino-uno-rev3 (accessed May 16, 2023).
[27] "NodeMCU V3 Lua CH340G ESP8266 Development Board F5D3," *ELKIM*, Jun. 24, 2021. https://www.elkim.no/produkt/nodemcu/ (accessed May 16, 2023).
[28] "ESP32D WROOM NodeMCU WiFi Bluetooth-utviklingskort med 38 pinner og CP2102-seriell," *ELKIM*, Dec. 05, 2022. https://www.elkim.no/produkt/esp32-wroom-32d-ubs-38p-cp2102/ (accessed May 16, 2023).
[29] "AX1500 Wi-Fi 6 Router." https://www.tp-link.com/no/home-networking/wifi-router/archer-ax1500/ (accessed May 16, 2023).
[30] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, May 1974, doi: 10.1109/TCOM.1974.1092259.
[31] "Evolution of HTTP | MDN," Apr. 10, 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP (accessed May 16, 2023).
[32] I. Craggs, "MQTT Vs. HTTP: Which protocol is the best for IoT or IIoT?" https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/ (accessed May 16, 2023).
[33] M. Sama, "HTTP Polling and Long Polling," *Cache Me Out*, Sep. 30, 2021. https://medium.com/cache-me-out/http-polling-and-long-polling-bd3f662a14f (accessed May 16, 2023).
[34] "HTTP request methods - HTTP | MDN," Apr. 10, 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (accessed May 16, 2023).
[35] "MQTT - The Standard for IoT Messaging." https://mqtt.org/ (accessed May 16, 2023).
[36] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, Eds., "MQTT Version 5.0." OASIS, Mar. 07, 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html
[37] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, May 2017, doi: 10.21105/joss.00265.
[38] W. Beaton, "Eclipse Mosquitto^{TM}," *projects.eclipse.org*, Nov. 07, 2013. https://projects.eclipse.org/projects/iot.mosquitto (accessed May 16, 2023).
[39] "Eclipse Paho," *projects.eclipse.org*, Jan. 31, 2013. https://projects.eclipse.org/projects/iot.paho (accessed May 16, 2023).

[40] I. Craggs, “Eclipse Paho Downloads.” https://www.eclipse.org/paho/index.php?page=downloads.php (accessed May 16, 2023).
[41] R. Hall, “Top 5 Reasons You Should Use Svelte on Your Current Project Right Now.,” *Medium*, Jun. 17, 2019. https://medium.com/@arxpoetica/top-5-reasons-you-should-use-svelte-on-your-current-project-right-now-e2f6835e904f (accessed May 18, 2023).
[42] I. Today, “The Role of Data in Industry 4.0,” *Industry Today*, May 20, 2020. https://industrytoday.com/the-role-of-data-in-industry-4-0/ (accessed May 21, 2023).
[43] N. O’Leary, “PubSubClient - Arduino Reference.” https://reference.arduino.cc/reference/en/libraries/pubsubclient/ (accessed May 19, 2023).
[44] “FreeRTOS (Overview) - ESP32 - — ESP-IDF Programming Guide latest documentation.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html (accessed May 20, 2023).
[45] “Tasks and Co-routines [Getting Started],” *FreeRTOS*. https://www.freertos.org/taskandcr.html (accessed May 20, 2023).

Appendiks A Time schedule

https://docs.google.com/spreadsheets/d/1L3CfLtuvzoEQ1dTiEUT3I605dHm1-se6/edit?usp=sharing&ouid=103996409854430007286&rtpof=true&sd=true

Appendiks B Components, tools, and other accessories

Can be viewed in the file named "Components, tools, and other accessories.zip"

Appendiks C Project repository

https://github.com/Baulind/ModelRailwayControll

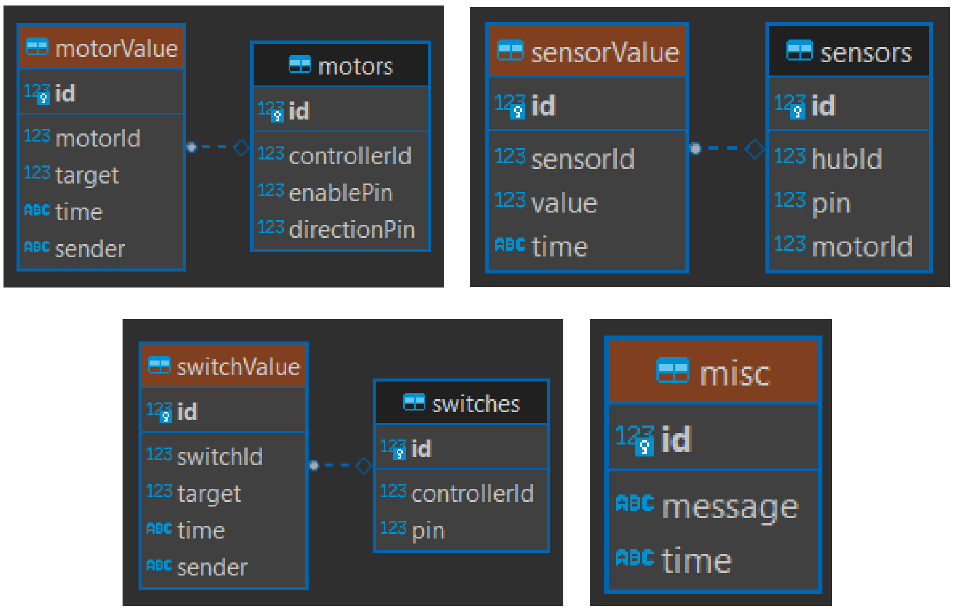
Appendiks D MQTT broker configuration

Listener	1883
Listener	9001
Protocol	websockets
socket_domain	ipv4
allow_anonymous	true

Appendiks E Database

E.1 ER Diagram

Miscellaneous logs. all nonstandard messages in base64 encoding such that the raw JSON string can be viewed later.

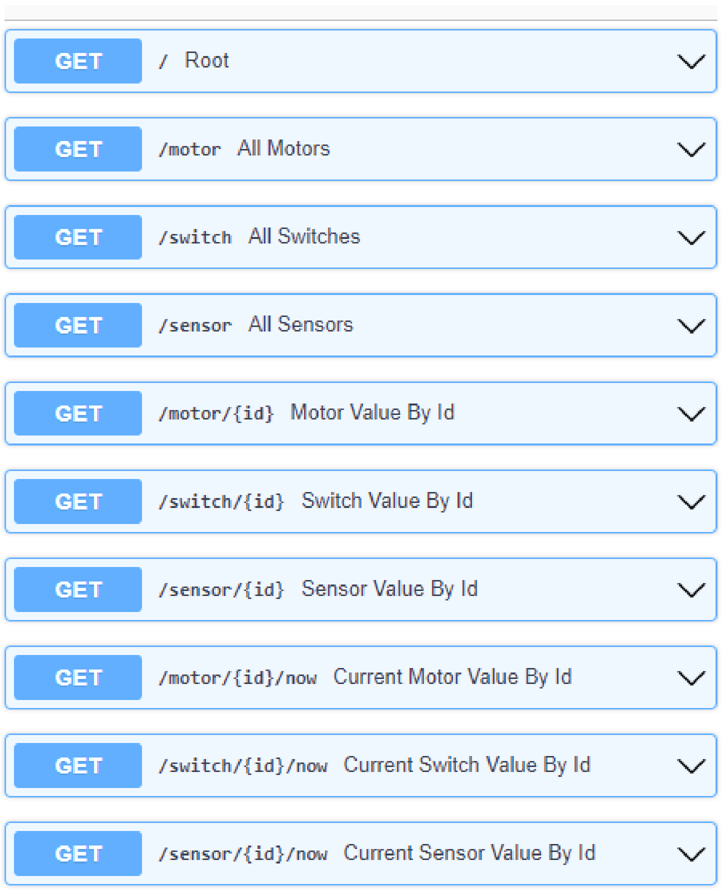


E.2 SQL Code

Found in project repository under Database\schema.SQL

Appendiks F API endpoints

Generated with the reserved /docs endpoint.



```
GET  /  Root
GET  /motor  All Motors
GET  /switch  All Switches
GET  /sensor  All Sensors
GET  /motor/{id}  Motor Value By Id
GET  /switch/{id}  Switch Value By Id
GET  /sensor/{id}  Sensor Value By Id
GET  /motor/{id}/now  Current Motor Value By Id
GET  /switch/{id}/now  Current Switch Value By Id
GET  /sensor/{id}/now  Current Sensor Value By Id
```

Appendiks G MQTT payload examples

Motor control:

{"Id":5,"Speed":100,"Sender":"96291786819"}

Configure motor pins:

{"Id":0,"Command":"setPins","Value":[1,2]}

{"Id":1,"Command":"setPins","Value":[3,4]}

{"Id":2,"Command":"setPins","Value":[5,6]}

Switch control:

{"Id":5,"State":false,"Sender":"96291786819"}

Sensor event:

{"Id":20,"Value":false,"Sender":"trackoccupancyhub0"}