

Edgar Andersen
Vanessa Berg
Larry Cheng
Hong-Ngan Hua
Ida Brattøy Ustad
Thomas Qvidahl

Verdens mest avanserte modelljernbane

Et sommerprosjekt om utbygging av modelljernbane med integrasjon av avansert sensorikk og teknologi



Prosjekteier: Elisabeth Cabrinetti
Prosjektleder: Hong-Ngan Hua
August 2025

Norges teknisk-naturvitenskaplige universitet - NTNU
Institutt for bygg- og miljøteknikk

Sammendrag

I dette sommerprosjektet har seks studenter med ulike fagbakgrunn innen konstruksjon, elektro, kybernetikk, IT og informatikk samarbeidet om å utvikle en avansert modelljernbane i H0-skala (1:87). Prosjektet er tverrfaglig og kombinerer landskapsutvikling inspirert av norsk natur med moderne teknologi. Det er installert et automatiseringssystem basert på Roco-komponenter, og det er utviklet sensorløsninger for planovergang, brovibrasjon, bevegelse og rundetelling, sistnevnte vist på en OLED-skjerm. Sensorene styres av ESP32-enheter og kommuniserer via RabbitMQ, med en Raspberry Pi som sentral node. Det er også gjort forberedelser for å implementere en hurtigklokke, døgnsykluser via prosjektor og LED-strips, som vil kunne simulere naturlig lysforløp. Flere av disse funksjonene er under utvikling og planlegges videreført i fremtidige prosjekt.

Figurer

1	Pinout referanse for ESP32 D1 mini.	6
2	Pinout referanse for SG90 servomotor.	7
3	Pinout referanse for 0.96-inch OLED-skjerm med 4 pinner.	8
4	Multimaus fjernkontroller til z21	8
5	Hvit z21-start avbildet til venstre og Z21 til høyre	9
6	2-dimensjonal plantegning av modelljernbanen	13
7	Manuell og automatisert sporveksel	15
8	Isolerende lasker	16
9	Påmontert isolerende lask mellom to blokker og rød ledning.	17
10	Lyssignal	18
11	Planovergang, slik kan det også gjøres...	21
12	Monteringsplate med ben og avstandsstykker for servomontering.	22
13	3D modell av bommene, klar til print	22
14	Ferdig planovergang, klar til montering	22
15	Planovergangsdekke, ferdig malt	23
16	GD-03B streklapp	23
17	Hurtigklokke klient	25
18	Testing av styrt lyslenke	25
19	Konstruksjon av fjell	30
20	Brohode	31
21	Forlatt gruve	31
22	Autovern	32
23	Test epoksy uten farge	33
24	Test epoksy innblandet blå farge	33
25	Bunnen i vannbassenget blir til	34

Tabeller

1	Elektroniske komponenter brukt i prosjektet	2
2	Modelljernbane-teknologi brukt i prosjektet	3
3	Jernbaneskinner brukt i prosjektet	3
4	Bygge- og landskapsutstyr.	4
5	Komponenter med åpen kildekode lisens	5
6	Liste over sporvekseladresser	16

7	Liste over filer for automatisk togkjøring	21
8	Komponentverdier	23

Innhold

Figurer	ii
Tabeller	ii
1 Innledning	1
2 Utstysrliste	2
2.1 Elektroniske komponenter	2
2.2 Modelljernbane-teknologi	3
2.3 Roco Geoline modelljernbaneskiner	3
2.4 Landskap	3
2.5 Brukernavn, passord og andre viktige koder	4
2.5.1 Aktiveringskode for wifi-utvidelsespakke (10814)	4
2.6 Ruterinformasjon	4
2.6.1 Raspberry Pi	4
2.6.2 RabbitMQ	5
2.6.3 Open Source komponenter	5
3 Teori	6
3.1 Elektroniske komponenter	6
3.1.1 Raspberry Pi	6
3.1.2 ESP32	6
3.1.3 Streklapp	7
3.1.4 Servomotor	7
3.1.5 OLED-skjerm	7
3.2 Modelljernbane-teknologi	7
3.2.1 Styringssystem for modelljernbane	7
3.2.2 Z21 applikasjoner	8
3.2.3 Java Model Railroad Interface (JMRI)	9
3.2.4 Blokkdeteksjon og oppdeling i blokker	9
3.2.5 Sporveksler	10
4 RabbitMQ - meldingsbuss	11
4.1 Installasjon av RabbitMQ på RPI	11
4.1.1 Aktivering av administrasjonsgrensesnitt og sikkerhet	11
4.2 Oversikt over topics	12

5	Design og oppsett av modelljernbanen	13
5.1	Jernbanen	13
5.2	Underlag og stigning	13
5.3	Svingradius	14
5.4	Ballast	14
5.5	Lokomotiv, vogner og CV-adresser	14
5.6	Sporvekslere, programmering og adresser	15
5.7	Togdeteksjon og blokkstyring	15
5.7.1	Valg av deteksjonsmetode	16
5.7.2	Fysisk oppbygning	17
5.7.3	Signalisering og videre bruk	17
5.8	Elektronikkinstallasjon	18
5.8.1	Strømforsyning 230V AC	18
5.8.2	Strømforsyning tog, 18V AC	18
5.8.3	Strømforsyning mikrokontrollere 5V DC	19
6	Moduler og metode for automasjon og sensorikk	20
6.1	Automatisert togkjøring (kode for tog)	20
6.1.1	Frontend	20
6.1.2	Backend	20
6.1.3	Implementering av sensorikk	20
6.1.4	Gjenstår	20
6.2	Planovergang	20
6.2.1	Gjenstår	23
6.3	Strekklapp	23
6.4	Visning av rundeteller med OLED-skjerm	24
6.5	Hurtigklokke	24
6.6	Programmerbar belysning	25
6.7	Projisering av lys/effekter på landskap	26
6.7.1	Skygge og lys	27
6.7.2	Animasjon	27
6.7.3	Lydeffekter	27
6.7.4	Valg av projektor	27
7	Landskap	29
7.1	Planleggingsfase	29

7.1.1	Byggeplate	29
7.2	Utførelse	29
7.2.1	Landskap	29
7.2.2	Vann	29
8	Vedlikehold og drift	31
8.1	Feilsøking	31
9	Videreutvikling	32
9.1	Integrering av signallys	32
9.2	Andre utviklingsmuligheter	32

1 Innledning

Mangelen på nyutdannede innen jernbaneteknologi er en økende utfordring for samferdselssektoren. For å møte dette har Konnekt, et samarbeidsinitiativ mellom Statens Vegvesen og Jernbanedirektoratet, satt i gang tiltak for å styrke koblingen mellom utdanning og arbeidsliv, og samtidig gjøre jernbanerelaterte utdanninger mer attraktive for unge.

Som en del av denne satsningen ble det sommeren 2025 startet et prosjekt ved Norges teknisk-naturvitenskapelige universitet (NTNU) i Trondheim for å påbegynne utviklingen av verdens mest avanserte modelljernbane. Seks studenter med ulike fagbakgrunn innen konstruksjon, elektro, kybernetikk, IT og informatikk samarbeidet om å kombinere landskapsutvikling inspirert av norsk natur med moderne teknologi. Modellen er utstyrt med et automatiseringssystem basert på Roco-komponenter og inneholder egenutviklede sensorløsninger for planovergang, brovibrasjon og rundetelling. Sensorene styres av ESP32-enheter og kommuniserer via RabbitMQ, med en Raspberry pi som hovednode.

I tillegg er det tilrettelagt og påbegynt arbeid med funksjoner som hurtigklokke, døgnsykluser via prosjektor og LED-strips for simulering av naturlig lysforløp. Prosjektet er planlagt som en flerårig satsing, der modellen gradvis bygges ut med mer avanserte funksjoner og brukes i undervisning på tvers av fagområder. Gjennom praktisk interaksjon får studentene direkte erfaring, noe som konkretiserer teoretiske konsepter, samt bidrar til å vekke og opprettholde interessen for jernbaneteknologi.

2 Utstysrliste

Gjennom prosjektet er det benyttet et bredt utvalg av utstyr og komponenter. Dette kapittelet gir en overordnet gjennomgang av disse. For bedre struktur er utstyret inndelt i ulike underkategorier.

I Underkapitlene 2.1 og 2.2 er de fleste komponentnavnene klikkbare lenker som fører direkte til de respektive kjøpsidene.

2.1 Elektroniske komponenter

Tabell 1: Elektroniske komponenter brukt i prosjektet

Komponent	Antall	Bruksområde	Kommentar
Raspberry Pi Model 4B 8GB	1	Hovedenhet for kommunikasjon, styring og data-behandling	Krever kjøling
MicroSD 32GB A1	1	Brukes til Raspberry Pi	Bør være av typen A1/A2
Joy-it Robust aluminiumskabinett for Raspberry Pi 4 B	1	Beskytter Raspberry Pi mot støt og overopphe-ting	
ESP32 D1 mini	1	Datatinnhenting fra sensorikk og styring av ak-tuatorer	Pinnene må loddes på kortet
Servomotor SG90	1	For styring av planover-gangen	SG90 har plastgir, og er dermed ikke like robust som MG90S som har me-tallgir
0.96 inch OLED-skjerm (4-pin)	1	Fremvisning av antall kjørte tog-runder	H0-skala-vennlig
Strekklapp GD-03B	6	Måling av vibrasjon på bro	
Diverse kretskomponenter	-	Perfboard, motstander, kondensatorer, kabler ol.	
Strømforsyning 5V (DC) 90W	1	Forsyner ESP32'er og lys-lenke med 5V.	Beskyttet mot overbe-lastning, overspenning og kortslutning
WS2812B light strip 5m, 5V	1	Simulere naturlig lysfor-løp	

2.2 Modelljernbane-teknologi

Tabell 2: Modelljernbane-teknologi brukt i prosjektet

Komponent	Antall	Bruksområde	Kommentar
Hvit z21 start med WiFi-utvidelsespakke (10814)	1	For styring av togene	Wifimodulen må aktiveres med en opplåsningskode
Z21 Detektorer x8	2	For å dele opp jernbanen i blokker (totalt 8x2)	Hvit z21 start har ikke støtte for CAN, kunne derfor tatt en x16 Detektor
Motor for Geoline sporveksel	6	Motor for å drive bytte mellom de ulike posisjonene	
Dekoder for Geoline sporveksel	6	Tillater programmering for styring av motor	

2.3 Roco Geoline modelljernbaneskinner

I prosjektet er det tatt i bruk Roco Geoline-skinner. Disse er utstyrt med integrert ballast som ikke er konstruert for å kunne fjernes. Det er derfor avgjørende at alt tilleggsutstyr som anvendes i prosjektet, er fullt kompatibelt med Geoline-systemet.

No.	Image	Number	Name	Scale	Producer	Pcs.
1		61106	G800	HO	Roco	4
2		61110	G200	HO	Roco	2
3		61111	G185	HO	Roco	19
4		61122	R2	HO	Roco	6
5		61123	R3	HO	Roco	2
6		61124	R4	HO	Roco	4
7		61128	2GB	HO	Roco	11
8		61140	TL	HO	Roco	2
9		61141	TR	HO	Roco	1
10		61154	CTL	HO	Roco	2
11		61155	CTR	HO	Roco	1
Total:						54

Tabell 3: Jernbaneskinner brukt i prosjektet

2.4 Landskap

Tabellen viser utstyret som ble brukt til bygging og utforming av landskap.

Tabell 4: Bygge- og landskapsutstyr.

Komponent	Bruksområde	Kommentar
Bygging		
Plate	Underlag for skinnegang og landskap	Kryssfiner anbefales
Sparkel	Utjevning av overflater	Slipes etter tørking for jevn finish
Avispapir	Papir-mâché til terreng	Billig og lett formbart
Lim	Sammenføyning av materialer	Trelim, kontaktlim eller hobbylim
Isopor	Forming av terreng	Lett å skjære og bygge høyder
Diverse elektriske verktøy	Kapping, boring, sliping	Bruk verneutstyr ved arbeid
Epoxy/Resin	Brukes til vann	Bruk lik type epoxy, ikke bland!
PLA	3D-printede spesialdeler	Kan brukes til å printe alt fra broer til detaljer.
Dekor		
Maling	Farge på landskap, bygninger og detaljer	Akryl eller modellmaling anbefales
Trær/busker	Landskapsdetaljer	Kan kjøpes ferdig eller lages selv
H0-skala bygninger	Realistisk miljø rundt skinnegang	
Naturressurser (stein, sand, o.l.)	Realistiske terrengeffekter	Sikres for å unngå løs smuss
Figurer	Realistisk miljø	

2.5 Brukernavn, passord og andre viktige koder

Dette underkapittelet gir en oversikt over nødvendige autentiseringsopplysninger og andre sentrale koder som er benyttet i prosjektet.

2.5.1 Aktiveringskode for wifi-utvidelsespakke (10814)

10814 serienummer: 132721

Aktiveringskode: 1515-2299-4364

z21 serienummer: 279440

z21 upgrade unlock-code: 3876344936

2.6 Ruterinformasjon

Original SSID:

SSID: z21

Passord: 86981897

IP-adresse for konfigurering av ruterinnstillinger: 192.168.0.1 (Skriv dette inn i nettleser)

Passord: NTNUKonnekt25

2.6.1 Raspberry Pi

Brukernavn: konnekt

Passord: minidrone

Kjent IP-adresse: 192.168.0.104

2.6.2 RabbitMQ

Brukernavn: konnekt

Passord: minidrone

Kjent link: <http://192.168.0.104:15672/>

2.6.3 Open Source komponenter

Vi har benyttet en rekke programvare komponenter lisensiert som åpen kildekode og andre komponenter lisensiert som Creative Commons. Disse krever attribusjon og er listet i Tabell 5.

Tabell 5: Komponenter med åpen kildekode lisens

Komponent/ Modell navn	Mo-	Kreatør	Lisenstype	Kilde URL	Notat
Brohode		Ukjent	CC BY 4.0	Lenke	Brukt til brohoder i landskapet
LED light strip rail for display cabinets		tibmart	Creative Commons - Attribution	Lenke	Montering av LED stripe over modellen
Arduino		The Arduino project	Several	Lenke	Programmering av ESP32
Raspberry Pi		raspberrypi.org	Several	Lenke	Styresystemer
JMRI		jmri.org	GPL 2.0	Lenke	Programmering av tog
OpenSCAD		openscad.org	GPL 2.0	Lenke	Modellering av planovergang etc.
Godot Engine		godotengine.org	MIT License	Lenke	3D landskap
Blender		blender.org	GPL 2.0	Lenke	3D landskap

3 Teori

I dette kapittelet gjennomgås den nødvendige teoretiske bakgrunnskunnskapen. Formålet er å gi leseren et solid grunnlag for å forstå innholdet i Kapittel 4 *RabbitMQ – meldingsbuss* og Kapittel 6 *Moduler og metode*.

Kapittelet er delt i to hoveddeler: teori tilknyttet de elektroniske komponentene og teori tilknyttet modelljernbaneteknologien.

3.1 Elektroniske komponenter

3.1.1 Raspberry Pi

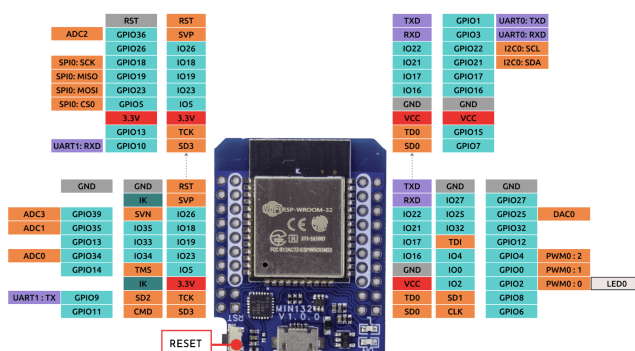
I prosjektet ble det benyttet en Raspberry Pi (RPI) som hovedenhet for kommunikasjon, styring og databehandling. RPI er en kompakt enkortsdatamaskin og kjører et Linux-basert operativsystem (Raspberry Pi OS). For å ta den i bruk kan man enten koble til skjerm, tastatur og mus, eller koble seg til via Secure Shell (SSH). Merk at dersom man ønsker å bruke SSH, må denne funksjonen aktiveres i systeminnstillingene til Raspberry Pi på forhånd.

Enheten leveres uten innebygd kjølesystem, noe som er viktig å være oppmerksom på. Ved belastende oppgaver kan prosessoren utvikle betydelig varme, og uten tilstrekkelig kjøling kan dette føre til termisk struping (throttling) eller ustabil drift. Det anbefales derfor å montere kjøleribber, vifter eller andre former for aktiv/passiv kjøling for å sikre stabil ytelse og forlenget levetid. For å sikre tilstrekkelig varmeavledning ble det i prosjektet benyttet et *Joy-IT robust aluminiumskabinett for Raspberry Pi 4B*, som gir både fysisk beskyttelse og fungerer som en effektiv kjøleløsning. Dette kabinettet er nevnt i utstyrstabell 2.1.

NB! Raspberry Pi støtter vanlig microSD-kort, men det anbefales å bruke A1/A2-klassifiserte kort. Disse er bedre egnet for applikasjoner som håndterer mange små *read/write*-operasjoner, og gir derfor raskere respons og bedre ytelse. Mange A1/A2 microSD-kort leveres ofte forhåndsinstallert med Raspberry Pi OS, noe som gjør dem *plug and play* og klare for bruk umiddelbart. Dersom operativsystemet ikke er forhåndsinstallert, kan det lastes ned gratis fra Raspberry Pi sin offisielle nettside og lastes opp på microSD-kortet.

3.1.2 ESP32

ESP32 er en kraftig mikrokontroller med innebygd støtte for både Wi-Fi og Bluetooth, og er godt egnet for IoT-applikasjoner. Den kan programmeres til å hente inn data fra sensorer eller styre ulike aktuatorer, og fungerer godt i et nettverk der flere enheter skal kommunisere trådløst.



Figur 1: Pinout referanse for ESP32 D1 mini.

I dette prosjektet ble ESP32 D1 mini valgt hovedsakelig på grunn av sin kompakte størrelse. Størrelsen gjør at enheten kan integreres i H0-skala vogner dersom det skulle monteres enkel sensorikk om bord i et togsett.

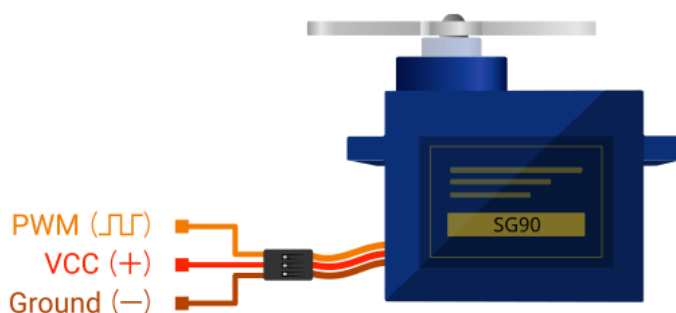
Det benyttes flere ESP32-enheter i nettverket, og disse kommuniserer med Raspberry Pi via en MQTT-meldingstjeneste, og er grundigere forklart i Kapittel 4. Hver ESP32 fungerer som en MQTT-klient som enten publiserer data fra sensorer eller abonnerer på styringsmeldinger fra Raspberry Pi.

3.1.3 Strekkklapp

En strekkklapp er en passiv komponent som (ofte) består av et piezoresistivt materiale, som gjør at motstanden i strekkklappen varierer med strekket som utføres på den. På denne måten kan man konvertere strekk til spenning, som kan måles med en ADC.

3.1.4 Servomotor

En aktuator som benyttes i prosjektet er en SG90-servomotor. Den styres ved hjelp av PWM-signaler (Pulse Width Modulation) og kan roteres til spesifikke vinkler innenfor sitt arbeidsområde. Servomotoren brukes her til å utføre mekaniske bevegelser knyttet til styring av planovergang.



Figur 2: Pinout referanse for SG90 servomotor.

3.1.5 OLED-skjerm

I prosjektet benyttes en 0.96-inch OLED-skjerm med I2C-grensesnitt for å vise antall runder hvert tog har kjørt. Skjermen har fire pinner, VCC for strømtilførsel, GND for jord, samt SCL og SDA for kommunikasjon.

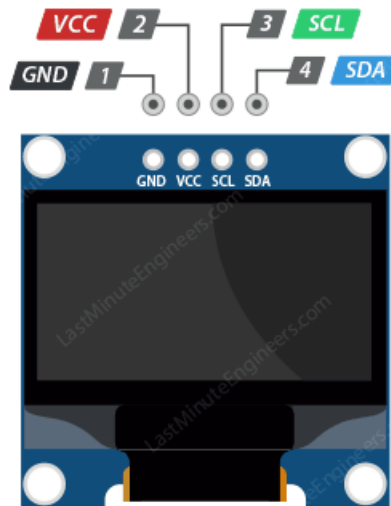
SCL (Serial Clock Line) fungerer som klokkesignalet og sørger for synkronisering av dataoverføringen mellom mikrokontroller og skjerm. SDA (Serial Data Line) er databussen som overfører selve informasjonen til skjermen. Disse to pinnene utgjør sammen et I2C-grensesnitt. Dette gjør det mulig å koble flere I2C-enheter til samme buss og styre dem med kun to datalinjer.

I prosjektet brukes det kun én ESP32 for å styre OLED-skjermen, og begge kommunikasjonspinnene er derfor koblet direkte til mikrokontrolleren.

3.2 Modelljernbane-teknologi

3.2.1 Styringssystem for modelljernbane

Styringssystemet som benyttes i prosjektet består av z21-start med WiFi-utvidelse, og var en del av et startsett sammen med lokomotiv og skinner. z21-start fungerer som styresentralen i



I2C OLED Display Pinout



Figur 3: Pinout referanse for 0.96-inch OLED-skjerm med 4 pinner.

nettverket og er tilkoblet togsinnene via en loddet ledning. Styringen kan skje enten ved hjelp av en Multimaus-fjernkontroll eller over nettverket via en smartenhet. Sistnevnte forutsetter at systemet er utvidet med en WiFi-utvidelse og at denne har blitt aktivert med opplæringskode i Z21 Maintenance Tool-appen.



Figur 4: Multimaus fjernkontroller til z21

Det finnes hovedsakelig to ulike styringsbokser fra produsenten: den hvite z21-start og den svarte Z21. Den svarte Z21-enheten finnes i flere varianter med ulik funksjonalitet, tilpasset ulike behov.

z21-start er en grunnleggende styringsenhet med begrenset funksjonalitet. I løpet av prosjektet ble det blant annet oppdaget at z21-start ikke har støtte for CAN-bus, noe som kan være en begrensning for mer avanserte styringssystemer og utvidelser.

For enkle formål og rimelig modelljernbanedrift er z21-start et kostnadseffektivt valg. Dersom man derimot ønsker økt funksjonalitet, inkludert WiFi-støtte uten ekstern modul, støtte for CAN-bus og mer avansert konfigurering, er den svarte Z21 et bedre alternativ.

3.2.2 Z21 applikasjoner

Z21-systemet støttes av flere applikasjoner som forenkler styring, konfigurering og vedlikehold av modelljernbanen. Dette inkluderer en Z21-app for smarttelefoner og nettbrett, samt Z21



Figur 5: Hvit z21-start avbildet til venstre og Z21 til høyre

Maintenance Tool for PC med Windows.

Z21-appen er tilgjengelig både i App Store og Google Play, og benyttes til å styre tog, bygge baner, programmere lokomotiver og kontrollere sporvekslere. Appen gir en intuitiv og brukervennlig grensesnitt for styring via smarttelefon eller nettbrett.

Z21 Maintenance Tool er et Windows-basert program som brukes til konfigurasjon og vedlikehold av alle Z21-komponenter, inkludert oppdateringer av firmware og systeminnstillinger. For brukere av z21-start med WiFi-utvidelse, må man aktivere WiFi-ruteren via dette verktøyet. Dette krever en opplåsningskode, som følger med startpakken eller fås via forhandler.

3.2.3 Java Model Railroad Interface (JMRI)

Java Model Railroad Interface (JMRI) er et open source programvaresystem for styring av modelljernbaner. Programmet er utviklet i Java og fungerer på tvers av plattformer. I tillegg består JMRI av flere applikasjoner med ulike bruksområder.

En sentral del av JMRI er PanelPro. Dette muliggjør design og å operere grafiske kontrollpaneler for modelljernbanen. Blant annet er det mulig å inkludere sporseksjoner, signaler, sporvekslere og sanntidsvisning av togposisjoner.

En annen viktig applikasjon er DecoderPro. Denne benyttes til programmering av digitale dekodere i lokomotiver, sporvekslere og annet utstyr. Denne applikasjonen forenkler prosessen med å endre CV-verdier (Configuration Variables) slik at det ikke er nødvendig å bruke en tradisjonell fjernkontroll som Multimaus.

JMRI er kompatibelt med en rekke digitale styringssystemer, inkludert Z21, og brukes ofte i kombinasjon med annen programvare og maskinvare for å utvide funksjonaliteten i modelljernbaneanlegg.

3.2.4 Blokkdeteksjon og oppdeling i blokker

Blokkdeteksjon er et prinsipp som brukes for å overvåke togbevegelser på en modelljernbane. Banen deles inn i flere blokker (soner), og det overvåkes om et tog befinner seg i en bestemt blokk. Når en blokk er opptatt, kan det forhindre andre tog i å kjøre inn i samme sone. Dette bidrar til å unngå kollisjoner og muliggjør mer avansert styring, for eksempel automatisk kjøreplan og dynamisk signalstyring.

Ved hjelp av blokkdeteksjon kan systemet sende statusmeldinger om hver blokk til en sentral styringsenhet. Disse meldingene kan for eksempel brukes til automatisering av togkjøring, styring av signallys, samt projisering av lys på selve banen.

I dette prosjektet brukes Z21 Detector-moduler for å gjennomføre blokkdeteksjonen. Hver Z21 Detector har flere uavhengige kanaler som kan overvåke egne blokker. To detektorer gir til sammen 16 kanaler, og dermed kan banen deles inn i inntil 16 blokker. Når et tog registreres i en blokk, markeres denne blokken som opptatt, og informasjonen sendes videre til systemet. I vårt prosjekt signaliseres blokkstatus videre til MQTT-meldingsbussen, som igjen formidler dette til enheter som styrer togkjøring, kjøreplaner og sensorikk. I tillegg lytter Godot på disse signalene

for å kunne projisere lysstatus direkte på banen.

Strømbasert deteksjon og Reed-kontakt er to ulike metoder for å oppnå blokkdeteksjon. Med strømbasert deteksjon føres kjørestømmen gjennom detektoren, slik at belastningen fra togets motor eller elektronikk utløser deteksjon. Dette krever at den ene skinnen i sporet er elektrisk isolert slik at strømmen til toget går gjennom detektormodulen. Dette oppnås ved å lage en liten glippe i sporet, eller ved å benytte seg av isolerende lasker mellom sporene. En magnetisk reed-bryter kan loddes mellom skinnen og detektoren. Når toget passerer og magnetfeltet fra toget aktiverer bryteren, detekteres toget.

3.2.5 Sporveksler

Sporvekslere leveres som mekaniske komponenter som i utgangspunktet må stilles manuelt for å endre spor. For å automatisere denne prosessen må sporveksleren utstyres med en vekselmotor, som fysisk flytter sporvekselen mellom posisjoner, og en dekode, som mottar digitale kommandoer fra styringssystemet og aktiverer motoren.

I et digitalt modelljernbanesystem tildeles sporvekslere egne adresser. Sporvekslene benytter en annen adresseliste enn lokomotivene. Lokomotivadresser brukes til kontinuerlig kontroll av hastighet og funksjoner, mens sporveksler adresseres som digitale funksjonsutganger, der signalet settes som "0" eller "1". Dette gjør at man kan kontrollere sporvekslere uavhengig av togstyringen, samtidig som alt styres via samme system, for eksempel med Z21, Multimaus eller programvare som JMRI.

Ved hjelp av denne løsningen kan sporvekslere integreres fullt ut i et digitalt styrt og eventuelt automatisert anlegg.

4 RabbitMQ - meldingsbuss

RabbitMQ fungerer som en meldingskø-tjeneste (message broker) og tillater kommunikasjon mellom ulike enheter i et nettverk. I IoT-nettverket tilknyttet modelljernbanen fungerer Raspberry Pi (RPI) som host for meldingskø-tjenesten RabbitMQ. På RPI er RabbitMQ installert og konfigurert med MQTT-plugin, slik at enheten kan fungere som en MQTT-broker (host).

ESP32-enhetene i nettverket fungerer som MQTT-klienter (slaver). Disse er utstyrt med sensorer og aktuatorer, og er programmert til å enten publisere eller abonnere på relevante emner (topics).

Når en ESP32 publiserer sensordata sendes meldingen til RabbitMQ på RPI, som deretter håndterer dataen. Det er også ønskelig at dataen framvises i et brukergrensesnitt, men slik funksjonalitet er foreløpig ikke implementert. Et eksempel er strekksensoren, som måler mekanisk tøyning og publiserer dataen i en topic. Deretter kan RPI abonnere på samme topic og behandle dataen, for så å eksempelvis illustrere dataen som en graf i et brukergrensesnitt.

Når en ESP32 abonnerer på en topic, henter den ut informasjon som en annen enhet i nettverket har publisert til den samme topicen. Et eksempel er planovergangen. RPI publiserer i en topic dersom et av togene nærmer seg planovergangen. ESP32-en tilknyttet planovergangen vil dermed aktivt lytte på denne topicen, og når meldingen mottas, aktiveres en servomotor som gradvis lukker planovergangen.

God meldingskontroll er essensielt for ethvert IoT-nettverk. Et velfungerende MQTT-oppsett gir et solid fundament for et system som skal være pålitelig, fleksibelt, modulært og skalerbart. Når MQTT-oppsettet er på plass, er det enkelt å videreutvikle systemet og koble opp flere mikrokontrollere.

Det finnes flere MQTT-brokere å velge mellom. I dette prosjektet er det tatt et bevisst valg om å bruke RabbitMQ fremfor Mosquitto. Denne vurderingen ble gjort på grunnlag av at RabbitMQ gir bedre støtte for videre integrasjoner og mer avansert køhåndtering dersom systemet utvides i fremtiden.

4.1 Installasjon av RabbitMQ på RPI

Følgende kommandoer skal kjøres i terminalvinduet på Raspberry Pi for å installere og sette opp RabbitMQ som MQTT-broker på port 1883:

- `sudo apt update` – Oppdaterer RPI
- `sudo apt-get install rabbitmq-server` – Installerer RabbitMQ
- `sudo systemctl enable rabbitmq-server` – Aktiverer automatisk oppstart
- `sudo systemctl start rabbitmq-server` – Starter RabbitMQ med en gang

4.1.1 Aktivering av administrasjonsgrensesnitt og sikkerhet

- `sudo rabbitmq-plugins enable rabbitmq-management`

For å få tilgang til RabbitMQ sitt webbaserte administrasjonsgrensesnitt, må følgende plugin aktiveres. Grensesnitt blir tilgjengelig på: `http://<RPI_IP-ADDRESS>:15672/`

Som standard finnes det en gjestebroker (brukeravn: **guest**, passord: **guest**) med begrenset sikkerhet. For økt sikkerhet anbefales det å slette denne og opprette egne brukere med riktige rettigheter:

- `sudo rabbitmqctl add_user username password`

-
- `sudo rabbitmqctl set_user_tags username administrator`
 - `sudo rabbitmqctl set_permissions -p / username ".*.*.*"`

For å slette gjestebrukeren:

- `sudo rabbitmqctl delete_user guest`
- `sudo rabbitmq-server restart`
- `sudo rabbitmqctl list_users`

4.2 Oversikt over topics

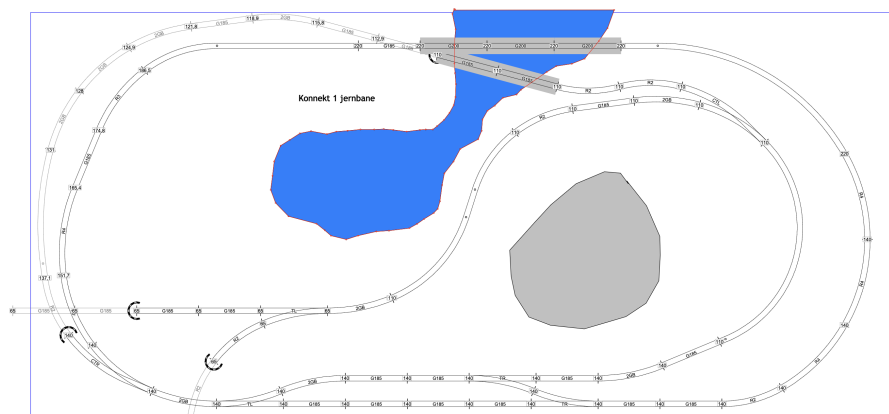
Topic	Publisher	Subscriber	Tilknyttet sensor / Kommentar
train/crossing	Raspberry Pi	esp32-servo	Servomotor for plan- overgangen
oled/obb	Raspberry Pi	esp32-oled	Teller for Öbb-toget
oled/cargonet	Raspberry Pi	esp32-oled	Teller for CargoNet- toget
lavbro/strekklapp _l	esp32-LavBro	Raspberry Pi	Venstre strekkklapp på høyeste broen.
lavbro/strekklapp _m	esp32-LavBro	Raspberry Pi	Midterste strekkklapp på høyeste broen.
lavbro/strekklapp _r	esp32-LavBro	Raspberry Pi	Høyre strekkklapp på høyeste broen.
hoybro/strekklapp _l	esp32-HøyBro	Raspberry Pi	Venstre strekkklapp på høyeste broen.
hoybro/strekklapp _m	esp32-HøyBro	Raspberry Pi	Midterste strekkklapp på høyeste broen.
hoybro/strekklapp _r	esp32-HøyBro	Raspberry Pi	Høyre strekkklapp på høyeste broen.

5 Design og oppsett av modelljernbanen

Banen er bygd opp på en kryssfinerplate på 240×120 cm. Applikasjonen SCARM ble benyttet for å designe modelljernbanen med korrekte skinner banelementer. Dette sikret at designet faktisk log seg bygge, og ga bedre oversikt over svingradius, stigninger, mengden skinner og plassering av elementer som broer og tunneller.

Under design og konstruksjon ble det tatt hensyn til flere tekniske og visuelle krav:

- Stigningen på banen skal ikke overstige 4%. For å sikre dette ble høydeforskjellene beregnet nøye, og det ble brukt isoporprofiler med 4% stigning som underlag.
- Minste tillatte svingradius er satt til $22,5^\circ$, for å etablere både realistiske og kjørbare svinger.
- Utforme et norsk-inspirert landskap, med illusjon av variert terreng. Nullhøyde er hevet slik at stigningen kan skje i begge retninger.
- Modulære løsninger for å imøtekomme ønsket om fleksibilitet og fremtidig ekspansjon.
- Inkludere minst én bro og én tunnel, samt plass til fjell, vannelement, steinbrudd, stasjon og bygninger.



Figur 6: 2-dimensjonal plantegning av modelljernbanen

5.1 Jernbanen

Utformingen av jernbanen har vært en iterativ prosess der det har blitt testet ulike teknikker og materialer for å finne gode løsninger, både konstruksjonsmessig og teknologisk. Valget av Geoline-skinner viste seg å være uheldig i vårt tilfelle. Selv om de er godt egnet for enkel oppbygging uten modifikasjoner, ble de lite fleksible når vi ønsket å tilpasse skinnene med lodding, blokkdeteksjon og broer. Det skapte flere utfordringer underveis, særlig i kombinasjon med ønsket om høy teknisk funksjonalitet.

5.2 Underlag og stigning

Banens underlag består hovedsakelig av isopor, valgt på grunn av lav vekt og høy formbarhet. Isoporen bearbeides med varmsag eller kniv, og gir et godt fundament for både skinnegang og landskapselementer. Stigninger er integrert ved hjelp av skråskjærte isoporprofiler med nøyaktig 4% helning, som oppfyller realistiske krav for modelljernbaner.

5.3 Svingradius

Med Geoline-skinner har vi i hovedsak fire typer kurver vi kan benytte:

- **R2:** 358mm, 30°
- **R3:** 443,5mm, 30°
- **R4:** 511mm, 30°
- **2GB:** 502,7mm, 22,5°

I tillegg benyttes fleksiskinner for å tilpasse individuelle kurver, men det er likevel viktig å unngå for krappe svinger. Det er også brukt sporveksler med både 22,5° og 30°. Ved tunnelbygging er det spesielt viktig å kontrollere at lengre passasjervogner passerer uten å berøre veggene, og vi har derfor unngått krappe svinger rett ved tunnelåpninger der det er mulig.

5.4 Ballast

For å etterligne realistisk jernbaneoppbygging har vi testet ulike metoder for ballast. Blant annet har vi brukt en blanding av sparkelmasse og sand for å sementere ned fleksiskinnene. Geoline-skinner kommer med støpt plastballast, men gir ikke samme visuelle kvalitet.

Vi har også eksperimentert med tradisjonell ballastering: sand og grus strøs over skinnene, børstes på plass og limes med en blanding av trelim og vann. Denne sprøytes med såpevann (vann + Zalo) for å bryte overflatespenningen slik at limet trekker ned i materialet. Vi har tilgjengelig flere typer sand og grus, og vil teste hvilken kombinasjon som gir best visuell og funksjonell effekt i slutfasen.

5.5 Lokomotiv, vogner og CV-adresser

I prosjektet benyttes to ulike lokomotiver: ett fra ÖBB (Österreichische Bundesbahnen) og ett fra CargoNet.

ÖBB-lokomotivet fulgte med i startsettet og har grunnleggende funksjonalitet. Det kan styres i begge retninger og har støtte for av/på-bryter av frontlys. Lokomotivet er ikke levert med lyddekoder, og har derfor ingen innebygde lydeffekter. Det er imidlertid mulig å ettermontere en lyddekoder dersom man ønsker å oppgradere funksjonaliteten. Dette krever kompatibel dekoder og eventuelle tilpassinger i kabinettet. **Lokomotivet er programmert med CV-adresse 1.**

CargoNet-lokomotivet ble anskaffet separat og tilbyr mer avansert funksjonalitet. Det er utstyrt med en lyddekoder som gir mulighet for avspilling av realistiske lydeffekter, som motorlyd, signalhorn og bremses. Lokomotivet leveres med flere forhåndsprogrammerte lyder, men ved hjelp av programvaren JMRI er det også mulig å laste opp egendefinerte lyder og justere lydprofiler for å tilpasse opplevelsen ytterligere. **Lokomotivet er programmert med CV-adresse 2.**

Prosjektet benytter i tillegg et utvalg vogner, inkludert både passasjervogner og ulike typer godsvogner, som åpne og lukkede containervogner. Disse bidrar til variasjon i togsettene og brukes til å simulere ulike togtransport-scenarier.

Dersom en lokomotivadresse (CV-adresse) skulle endres eller ukjent adresse er registrert, finnes det en metode for å identifisere hvilken adresse et lokomotiv er programmert med. Det er da viktig å sikre at kun ett lokomotiv står på sporet av gangen.

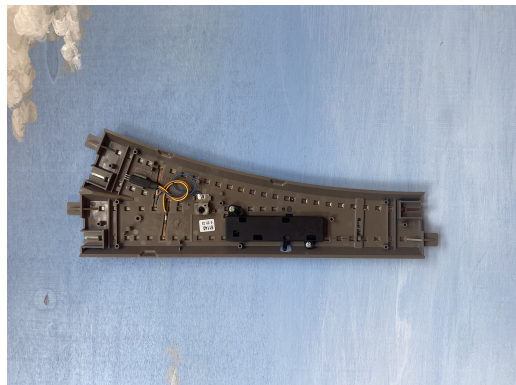
Begge lokomotivene fjernes først fra sporet. Deretter plasseres ett lokomotiv tilbake på sporet, og funksjonen for "Identifiser dekoder" benyttes, enten via DecoderPro i JMRI eller gjennom Z21-appen. Dersom det er korrekt kontakt mellom sporet og lokomotivet, vil lokomotivet bevege

seg litt frem og tilbake. Dette er et tegn på at det leses av informasjon fra dekodeeren. Det er viktig at lokomotivet får fullføre denne prosessen uten forstyrrelser.

Når lokomotivet er ferdig med avlesningen, vil den nåværende dekodeadressen vises på skjermen. Deretter kan lokomotivet fjernes fra sporet, og samme prosess gjentas for neste lokomotiv.

5.6 Sporvekslere, programmering og adresser

I dagens banelayout benyttes det seks sporvekslere. For å kunne styre disse automatisk må de utstyres med både motor og dekodeer. Sporvekslere uten motor må aktiveres manuelt, mens motoriserte sporvekslere med dekodeer kan styres digitalt via kontrollsystemet.



(a) Manuell sporveksel



(b) Sporveksel utstyrt med motor og dekodeer

Figur 7: Manuell og automatisert sporveksel

For å sette en sporveksel i programmeringsmodus, må den raskt toggles manuelt fem ganger. Når dette er gjort, vil sporvekselen bevege seg fra side til side, én gang per sekund i 30 sekunder, som en bekreftelse på at den er i programmeringsmodus. Etter 30 sekunder avsluttes programmeringsmodusen automatisk.

For å tildele en adresse til sporvekseldekodeeren har vi benyttet Multimaus-fjernkontrollen. Multimaus må settes i sporvekselmodus, som skiller seg fra togstyringsmodus ved at adressene vises som Txxx, for eksempel T001. Det er viktig å merke seg at adresser for sporvekslere og lokomotiver tilhører to separate registre og derfor ikke overlapper.

Når fjernkontrollen er i riktig modus og sporvekselen er satt i programmeringsmodus, kan ønsket adresse tastes inn og bekreftes med OK. Sporvekselen lagrer da den nye adressen og går automatisk ut av programmeringsmodus.

For digital styring via programvaren JMRI benyttes PanelPro. Her går man inn på Tools > Tables > Turnouts for å få oversikt over registrerte sporvekslere. Dersom en sporveksel ikke allerede ligger i systemet, kan den legges til manuelt via "Addfunksjonen". Her angis kun tallverdien på adressen, uten bokstaven T. For eksempel: dersom sporvekselen er T001 i Multimaus, skrives kun tallet "1" i adressefeltet. I samme meny kan man også gi sporvekselen et kallenavn for enklere identifikasjon. Det er også mulig å bytte mellom sporvekslenes to posisjoner (0 og 1) ved å bruke Statefeltet.

5.7 Togdeteksjon og blokkstyring

For å muliggjøre automatisert togstyring og unngå kollisjoner, må det til enhver tid sendes informasjon om hvor toget befinner seg, og hvilke baner som er opptatt. Dette oppnås ved å dele modelljernbanen inn i isolerte blokker. Systemet vårt består benytter to Z21 x8 Detector-enheter fra Roco, som til sammen støtter opptil 16 deteksjonskanaler. Dette gir grunnlag for inndeling av banen i inntil 16 blokker.

Tabell 6: Liste over sporvekseladresser

Sporveksel-modell	Plassering	Adresse	Kommentar
		1	
		2	
		3	
		4	
		5	
		6	
		7	

5.7.1 Valg av deteksjonsmetode

Basert på produsentens dokumentasjon ble to deteksjonsmetoder vurdert:

1. **Strømbasert deteksjon:** Kjørestrømmen ledes gjennom detektoren ved å kutte én skinne og deretter lodde en ledning fra skinnen og til detektoren. Når et tog befinner seg i blokken, trekker det strøm og blokken registreres som opptatt.
2. **Reed-kontakt:** En reed-kontakt loddas mellom én skinne og detektoren. Når toget passerer, aktiveres kontakten av togets magnetfel, og detektoren registrerer dette.

Reed-kontakter ble testet i praksis, men ga ikke tilfredsstillende resultater. Magnetfeltet fra togene var for svakt til å aktivere kontaktene pålitelig med mindre disse ble plassert tett inntil togets underside. Undersøkelser viste at andre modelljernbaner benytter ekstra magneter på togene, noe som ikke var beskrevet i vår dokumentasjon. I tillegg gir reed-baserte deteksjonspunkter kun informasjon om én posisjon, og er derfor uegnet for blokkreservasjon som krever at hele blokken overvåkes.

På bakgrunn av dette ble strømbasert deteksjon valgt som løsning. Denne metoden gir kontinuerlig og robust indikasjon på togtilstedeværelse i hver blokk. Etter et besøk til Ringve ble det oppdaget at de benytter isolerende lasker, som vist i Figur 8, mellom skinnekoblingene. Disse gir en tilnærmet usynlig og lite inngripende måte å dele banen opp i blokker. Denne løsningen ble vurdert som den mest hensiktsmessige for vårt prosjekt, både med tanke på funksjonalitet og estetikk.

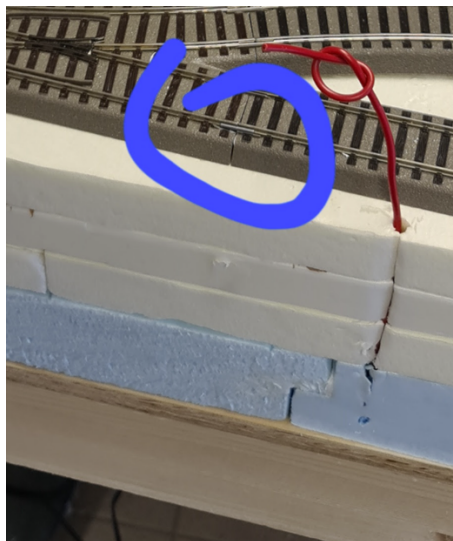


Figur 8: Isolerende lasker

5.7.2 Fysisk oppbygning

Den fysiske inndelingen av blokker gjøres ved å isolere én av skinnene mellom hver blokk ved hjelp av isolerende lasker. Siden det benyttes Geoline-skinner i prosjektet, medfører det noe ekstra arbeid for å tilrettelegge loddepunkter. Det må freses ut plast for å gi plass til loddepunktet, og det er utfordrende å unngå at plasten smelter. Etter testing har vi utviklet en metode som gir et nesten usynlig resultat sett fra utsiden.

Fra hvert loddepunkt bores det ned gjennom bordplaten, og det benyttes 0,5 mm² fleksibel rød ledning for å føre signalet ned til blokkdetektoren som er montert under bordet. Kjørestrømmen til toget går nå gjennom detektoren, som registrerer strømgjennomgang og markerer blokken som opptatt.



Figur 9: Påmontert isolerende lask mellom to blokker og rød ledning.

Blokkdetektorene kommuniserer med Z21-sentralen via Roco R-Bus. Når en blokk registreres som opptatt, publiseres dette som en melding på MQTT-meldingsbussen, se kapittel 4. Abonnenter på MQTT, som lyssignaler, kjøreplanstyring og Godot-applikasjonen (som projiserer lyseffekter på banen), benytter disse signalene for å ta avgjørelser i sanntid.

5.7.3 Signalisering og videre bruk

Systemet er designet med tanke på modularitet, fleksibilitet og mulighet for fremtidig utvidelse med nye teknologier. Ved å implementere blokkdeteksjon legges det et viktig fundament for videre funksjonalitet, ettersom mange sensorer og styringssystemer er avhengige av kunnskap om hvor togene befinner seg på banen.

Et eksempel på slik videre bruk er implementering av lyssignaler langs sporet. Dette er ikke inkludert i første fase av prosjektet, men det er tilrettelagt for det gjennom blokkdeteksjon og publisering av blokkstatus via MQTT. Dermed kan slike systemer enkelt etterimplementeres.



Figur 10: Lyssignal

5.8 Elektronikkinstallasjon

Det meste av elektronikken er installert under bordet for å holde overflaten ryddig og visuelt tilfredsstillende.

En WiFi-ruter er tilkoblet Z21-sentralen og gir dermed både internettforbindelse samt et lukket lokalt nettverk. Dette nettverket benyttes av Raspberry Pi-en og ESP32-mikrokontrollerne for kommunikasjon og styring, blant annet for tjenester som hurtigklokke og MQTT. Dette muliggjør fremtidig utvidelse av systemet med mer avansert styring og visualisering.

Det gjenstår å teste om Raspberry Pi også har kapasitet til å kjøre Godot-applikasjonen for prosjeksjonsmapping.

Z21-sentralen kommuniserer med dekodeerne (D1, D2) via R-BUS, og har i tillegg egen tilkobling for MultiMAUS-fjernkontroll via X-BUS.

5.8.1 Strømforsyning 230V AC

For tilførsel av 230V til hele anlegget har vi montert en seks-veis skjøteblokk under bordet. Dette gjør det mulig å koble til mye av den nødvendige elektronikken gjennom ett felles strømuttak.

5.8.2 Strømforsyning tog, 18V AC

Z21-sentralen forsyner togene med 18V vekselspanning (AC) via skinnene, levert på terminalene merket P og N. DCC-signalet for digital styring sendes også ut via disse terminalene. Sentralen får strøm fra en 2A adapter, noe som potensielt kan bli en begrensning ved større utvidelser.

Strøm fra 18V-uttaket føres videre til begge dekodeerne (D1, D2). Disse er koblet slik at P-terminalen går til den ene skinnen, mens N-terminalen kobles kontinuerlig til den andre skinnen rundt hele banen. **Det er viktig derfor at man er konsekvent med hvilken skinne som isoleres ved blokkdeteksjon.**

5.8.3 Strømforsyning mikrokontrollere 5V DC

ESP32-kontrollerne, Raspberry Pi-en og LED-lenken forsynes med 5V likespenning (DC) fra en felles 90W strømforsyning. Dette sikrer stabil drift for alle lavspente komponenter i systemet.

6 Moduler og metode for automasjon og sensorikk

I dette kapittelet presenteres de ulike modulene som er utviklet for automasjon og sensorstyring i prosjektet. Hver modul beskrives med hensyn til funksjon, implementeringsmetode og samspill med øvrige systemkomponenter. For å gjøre det enkelt å navigere til kildekoden, er flere av overskriftene i kapittelet klikkbare lenker som peker direkte til de respektive modulene i prosjektets GitHub-side.

6.1 Automatisert togkjøring (kode for tog)

Ved implementasjon av automatisert togkjøring er målet å ha et komplett autonomt system hvor tog kan vike for hverandre, regulere hastighet, sette stopp, og som i fremtiden enkelt kan kobles opp til andre systemer. Koden er skrevet i python.

6.1.1 Frontend

Frontend for debugging er et bilde av layouten til modelljernbanen, hvor banen er klippet ut, og polygons plassert bak bildet skifter farge for å definere hvilke blokker som er okkupert av hvilke tog.

Ved kjøring av main.py vil man få muligheten til å velge debugmodus, eller vanlig. I debugmodus får man muligheten til å tegne nye polygons. Plasser musa på ønsket posisjon for polygon-hjørne, og venstreklikk for å kopiere koordinatene til utklippstavlen. Legg deretter til i blocks.py. I vanlig modus vil bildet ikke ha grid og mulighet til å kopiere koordinater. I begge moduser legger man til tog, og deres startblokk i terminal.

6.1.2 Backend

Logikken til programmet foregår i main.py. Her bestemmes hvor toget skal flyttes, vikeplikt og stans. Fra kjøring av main.py går programmet følgende:

1. Terminalen tar inn data.
2. reader.py leser koblingene av sporene fra layout.txt, og kobler opp de til blokkene i blocks.py.
3. map.py tegner kartet med polygons bak.
4. main.py kjører logikken og oppdaterer okkuperte blokker ved å endre dataen i blocks.py.

Kommentarer er også før inne i koden.

6.1.3 Implementering av sensorikk

6.1.4 Gjenstår

6.2 Planovergang

Vi ønsket å lage en planovergang (noe lignende som figur 11) med bommer som går opp og ned, og koble denne til sensorikken slik at bommen automatisk går igjen når et tog nærmer seg.

En servo monteres midt under skinnegangen og har en arm som skyver/trekker motsatt vei på begge sider. Denne armen linkes med støtstang til en 90 graders arm slik at bevegelsen

Filnavn	Beskrivelse
block.py	Klasse for blokker.
blocks.py	Ordbok med posisjoner til polygons.
layout.txt	Fil som inneholder hvilke blokker som er koblet sammen. Kan redigeres, men polygons må da tegnes i henhold og kart må tilpasses.
main.py	Hovedprogram. Kjør i venv med <code>python main.py</code> for å starte.
map.py	Tegner kartet og inneholder verktøy for å forenkle prosessen med å tegne nye polygons.
mapFix.png	Utsnitt av kartet i PNG-format.
quick_test.py	Test for å sjekke at programmet fungerer.
reader.py	Leser data fra layout.txt og formaterer den for enklere tilgang.
test_run.py	Eksempel på en ekstra test.
train.py	Klasse for tog.

Tabell 7: Liste over filer for automatisk togkjøring

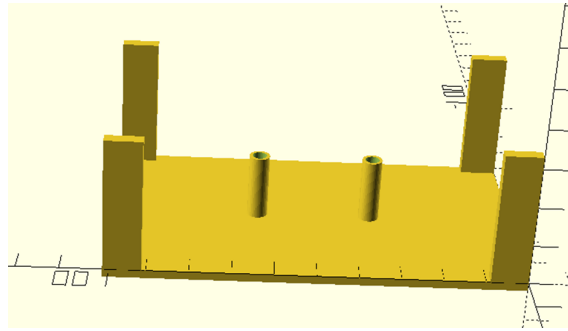


Figur 11: Planovergang, slik kan det også gjøres...

omdirigeres opp/ned. Denne linkes igjen med støtstang til en arm koblet til enden av bommen, som åpner/lukker bommen. Dette er ikke realistisk utført, men er ment som en demonstrasjon.

Planovergangens servomotor er koblet opp til en ESP32, med GPIO32, som lytter etter melding fra en topic i MQTT. Ved hjelp av togdeteksjon og toglogikk sendes det et høyt digitalt signal "1" til denne MQTT topicen når toget nærmer seg planovergangen. ESPen som lytter på topicen skal begynne å senke bommene når den leser "1". Når toget har passert planovergangen skal koden for toglogikk sende et lavt digitalt signal "0" til MQTT topicen. Når ESPen leser dette skal den heve bommen og den vil være åpen fram til neste gang det sendes et høyt digitalt signal. Når bommen er åpen settes vinkelen til 45°, og når den er lukket settes den til 0°.

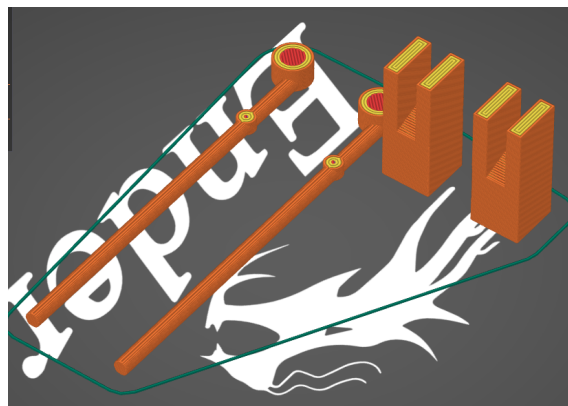
Grunnet lite og dyrt utvalg av planovergang har vi laget en selv. Her har vi modellert alle deler selv, med OpenSCAD, tegningene ligger her.



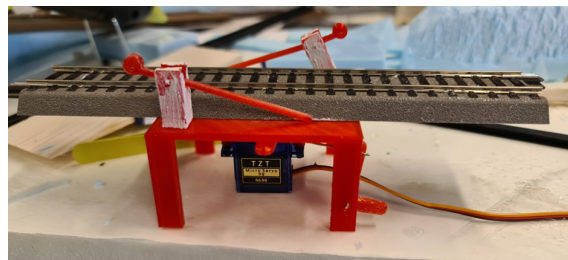
Figur 12: Monteringsplate med ben og avstandsstykker for servomontering.

Siden vår servo med arm bygger ca 35mm i høyden, blir det nødvendig å ta hull i underlaget der vi har planer om å montere den.

Støtstenger er utformet med bøyd binders, både fordi det var billig, lett tilgjengelig og godt egnet til formålet.



Figur 13: 3D modell av bommene, klar til print



Figur 14: Ferdig planovergang, klar til montering

Der veien krysser skinnene, må vi ha planovergangsdekke. Til det formålet fant vi en 3d-modell av treverk, som fikk ekstra realisme når vi pusset det med sandpapir og malte med brun/gul maling.



Figur 15: Planovergangsdekke, ferdig malt

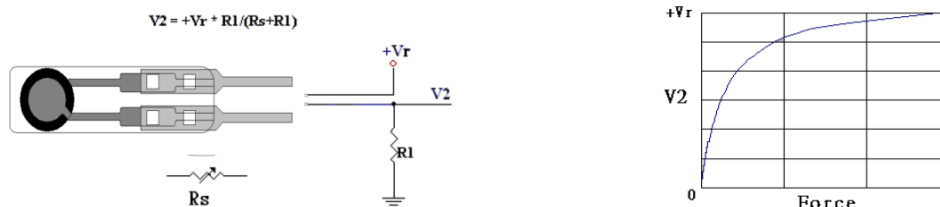
6.2.1 Gjenstår

- Maling og detaljering av planovergang
- Testing av ESP32 og software som styrer servo; skal på melding fra MQTT (blokkdeteksjon) aktivere bommen (Dette er testet og fungerer). Må tune vinkelen for servo for å få riktig posisjon på planovergang.

6.3 Streklapp

For måling av belastning i broseksjoner av modelljernbanen, er det blitt tatt i bruk streklapper av typen GD-03B. Streklappene er festet mellom bjelkene og togsykkene. Dataen sendes gjennom ESP32 på MQTT.

Kretsen for streklappen utgjøres av streklappen selv, en spenningsdeler, kondensator og ESP32 sin $3.3V$, GND og en av dens $ADC1$ porter ($IO32 - IO39$). Siden ESP32 bruker MQTT er det kritisk at man kun bruker $ADC1$ portene, som ikke påvirkes av at ESP32 er koblet på internett, i motsetning til $ADC2$ portene.



Figur 16: Kablingsskjema for krets til streklapp. Kondensator C_1 er ikke tegnet inn, men ligger over streklappen. Bildet er hentet fra databladet til GD-03B, linken kan finnes i utstyrslisten.

Komponent	Verdi	Kommentar
R_s	-	Type GD-03B
R_1	$1\text{ M}\Omega$	Anbefalt i datablad $1\text{ k}\Omega - 100\text{ k}\Omega$.
C_1	$224\text{ }\mu F$	Ligger over streklappen. Strengt tatt ikke nødvendig, men fjerner støy.
$+V_r$	3.3 V	Spennning fra ESP32
V_2	-	Målepunkt, kobles til en av ESP32 sine $ADC1$ porter.

Tabell 8: Komponentverdier

For å kunne bruke flere streklapper, er IO-portene 32 – 34 tatt i bruk på 2 ESP32-enheter, for å kunne måle fra 6 streklapper samtidig. Kretsen er i dag bygd slik at alle 6 streklapper drives av 3.3 V fra 1 ESP32, mens jord er koblet til begge ESP32 sin GND for å sikre samme referanse.

6.4 Visning av rundeteller med OLED-skjerm

For å gi en visuell indikasjon på hvor mange runder hvert tog har kjørt, benyttes en 0,96-tommers OLED-skjerm som er plassert i nærheten av stasjonsområdet. Skjermen drives av en ESP32 og er koblet til prosjektets MQTT-baserte kommunikasjonssystem.

Hver gang et tog detekteres i en blokk i nærheten av stasjonen, sendes en melding via MQTT som oppdaterer rundetallet for det aktuelle toget. Hvert tog har sin egen dedikerte topic i MQTT-strukturen, noe som sikrer at informasjonen holdes adskilt og korrekt. ESP32-enheten som styrer OLED-skjermen abonnerer på disse topicene og oppdaterer skjermbildet fortløpende med gjeldende rundetall.

OLED-skjermens kompakte størrelse gjør den godt egnet til bruk i modelljernbaneanlegget, da den integreres naturlig i landskapet uten å virke visuelt forstyrrende. Nettopp derfor ble 0,96-tommers varianten valgt. For å forbedre det estetiske uttrykket ytterligere, og for å skjule elektronikk og kabler, planlegges det å 3D-printe en spesialtilpasset kapsling til skjermen. Dette vil bidra til et mer helhetlig og ryddig visuelt inntrykk i modellen.

Tilkoblingspunkter for OLED-skjermen er som følger:

- VDD = 3,0V
- GND = GND
- SDA = GPIO 25
- SCL = GPIO26

6.5 Hurtigklokke

Under modelljernbaneklubbens kjørekvelder, opererer de med en tidsplan for kjøring av togene. Togtabellen har normale rutetider, og for å simulere tid som går “raskere enn normalt”, bruker de en hurtigklokke, der ett døgn feks kan gå på 3 timer. På den måten kan man kjøre gjennom alle togavganger til “normal” tid, uten å faktisk bruke et helt døgn på det.

Vi ønsket å ha vår egen hurtigklokke, som vi kunne lese av tiden fra, for å synkronisere ulike tjenester; dagslyssimulering, lyder i dioramaet (mennesker, hanegal etc), ulike aktiviteter og lyseffekter (skygge, animasjon av vann) projisert ned ovenfra med prosjektor (se egen systembeskrivelse).

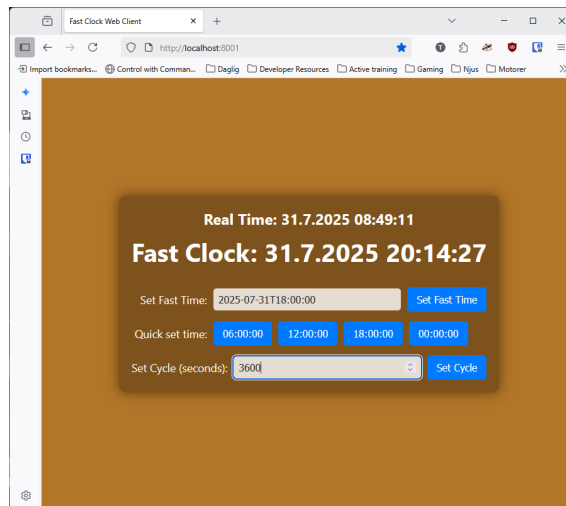
For å implementere dette, har vi skrevet et REST API som har to endepunkter; time og cycle. Time er hurtigklokkens nå-tidspunkt og cycle er hvor mange sekunder et simulert døgn tar. Eksempel; med cycle=86400 er 1 simulert døgn = 1 reelt døgn, for det er 86400 sekunder i døgnet. Med cycle = 3600 går døgnet unna på 1 reell klokke-time; 3600 sekunder i en time.

API'et er implementert i Python, med FastAPI og uvicorn og pakket i en Docker-container.

For å demonstrere bruk av API'et, samt å enkelt både kunne se og sette time/cycle, har vi også laget en klient i HTML/JS/CSS. Denne kan kjøres rett i en nettleser, men også serves med en Python HTTP server i en egen container, som gjør det mulig å åpne klienten fra en ren nettleser, feks iPad.

I vårt system kjører hurtigklokken på en Raspberry Pi 4B som er koblet til 2.412 WLAN, publiserer hurtigklokke-tid på et REST endepunkt.

Vi har gjort en vurdering av om vi skal publisere hurtigklokkens tid på MQTT. Siden tidspunktet beregnes for hver gang en klient sender en forespørsel, er API'et sovende"all tid det ikke sendes forespørsler. En kontinuerlig strøm av oppdateringer til MQTT vil føre til mangedobling i ressursforbruk på CPU, med tilhørende varmeutvikling, uten av man får noe spesiell gevinst av dette. I et større miljø med mange flere klienter og andre behov vil man kunne vurdere dette annerledes.



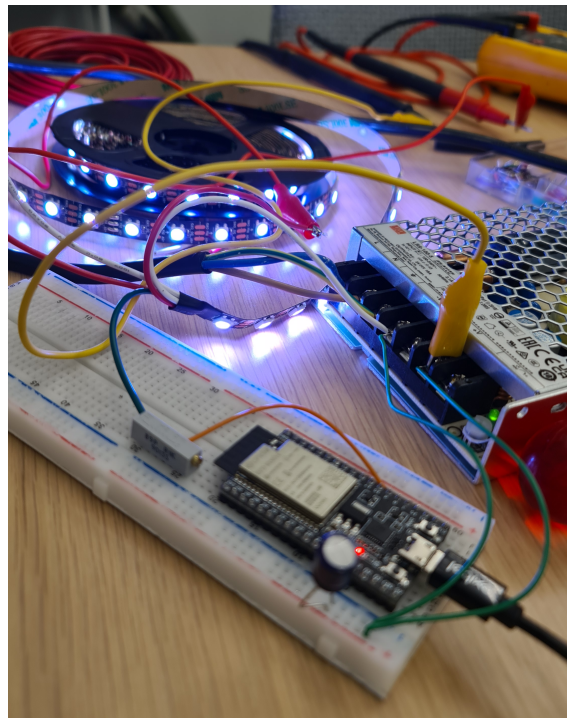
Figur 17: Hurtigklokke klient

Gjenstår:

- API'et kan beregne og publisere korrekt fargeverdi for tidspunkt/årstid, slik at alle klienter får koordinert fargeverdi.

6.6 Programmerbar belysning

For å skape ulike stemninger i landskapet benyttes en programmerbar LED-lyslenke styrt av en ESP32-mikrokontroller. Hensikten er å etterligne naturlige lysforhold som daggry, soloppgang og solnedgang, og dermed komplementere effekten av projisering i anlegget, se kap. 6.7.



Figur 18: Testing av styrt lyslenke

Under konfigurering av LED-lyslenken ble denne guiden fulgt. Basert på denne guiden ble kretsen

designet i henhold til skjematetegningen. Vi har loddet ned en ESP32 på et perfboard. En kondensator er lagt i parallel mellom 5V og GND for å motvirke spenningsfall, og en motstand i serie på datautgangen reduserer strømgjennomgang til LED-lenken.

Programvareoppsett:

- Logger seg på z21 nettverket (hardkodet ssid/passord)
- Kobler til hurtigklokke-api på adresse 192.168.0.107:8000 (hardkodet)
- Leser så hurtigklokke-tid 1 gang pr sekund, og transisjonerer fargeverdi på lysdiodene deretter.

Leverandøren Kjell & Company oppgir at hver LED kan trekke opp til 60mA, og at det er 60 LED pr meter lyslenke. Lyslenken er 5 meter.

$$60\text{ mA} \cdot 60\text{stk}/m \cdot 5m = 18\text{ A}$$

$$18\text{ A} \cdot 5\text{ V} = 84\text{ W}$$

Under testing har vi ikke observert belastning på mer enn 2.9A ved full belysning, og vi kan derfor bruke restkapasiteten til andre formål som krever 5V, feks ESP32.

Gjenstår:

- Dele lenken i to deler som styres separat. En del monteres over bakgrunnen (himmel) i 45° vinkel. Den andre delen bak fjellet der toget kjører, for å skape stemingsbelysning.
- Programmere lysstyringen av lenken slik at det mer pålitelig matcher hurtigklokkens tid/fargenyanse
- Programmere fargespill (lysdiodene er individuelt programmerbare) som etterligner soloppgang/nedgang.

6.7 Projisering av lys/effekter på landskap

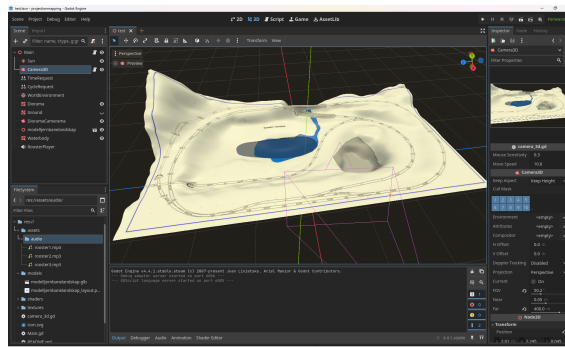
OBS! Dette er en modul som foreløpig er på test- og utviklingsstadiet.

Ved å montere en projektor i taket og peke den ned på dioramaet, kan vi simulere ulike lyseffekter på landskapet.

For å skape et miljø for dette, har vi benyttet spillmotoren Godot. Vi har laget en 3D-modell som en enkel mesh i Blender og lagt tegningen av jernbanesporet for referanse. Deretter kan vi forme landskapet så det stemmer med dioramaet vi bygger.

En preliminær modell av landskapet er laget i Blender og importert til Godot. For mer nøyaktig modellering, må man jobbe med det ferdige landskapet.

Vi kan alternativt benytte laserscanning for å skape en nøyaktig 3d-modell av landskapet.



(I den ferdige løsningen skal ikke teksturen med banelayout og "vann" være med, dette er kun for referanse under modelleringen.)

6.7.1 Skygge og lys

Vi har programmert en sol som beveger seg over himmelen i scenen, og dette skaper lys og skygge-effekt på landskapet. Solen henter tid og hastighet fra Hurtigklokke-API'et.

Der det er broer i landskapet, må vi legge inn en noe "i 3d-modellen, slik at broen kaster skygge.

Vi kan lese blokkdetektor-signaler og lyse opp jernbanen for å illustrere hvilke blokker som er aktivert, med feks rødt og grønt lys.

Forsøkt og forkastet: Refleksjon av bølger. Refleksjoner er avhengig av vinkel til lyskilden, så det får vi ikke til å simulere, siden retningen ikke stemmer for observatøren.

6.7.2 Animasjon

Vi kan animere ulike effekter på landskapet via projektor. Noen ideer:

- Krusninger på vannet
- Skygge av skyer som glir over landskapet
- Folk som går
- Biler som kjører
- Lys i hus og langs vei om natten

6.7.3 Lydeffekter

I tillegg har vi lagt til mulighet for lydeffekter i denne scenen, med hanegal som eksempel; et tilfeldig tidspunkt mellom kl 04 og 09 vil en av tre lydfiler med hanegal avspilles.

6.7.4 Valg av projektor

Vi har gjort noen enkle beregninger for å se hva slags prosjektor vi må ha for å kunne belyse hele dioramaet.

Throw ratio (kastforhold) for en prosjektor forteller hvor langt unna den må stå for å gi en viss bildestørrelse.

Throw ratio er gitt

$$ratio = \frac{avstand_til_lerret}{bildebredde}$$

derfor

$$avstand_til_lerret = ratio \cdot bildebredde$$

Med en bildestørrelse på $240 \times 120 \text{ cm}$ og dersom vi antar at vi kan plassere projektoren 2 meter over dioramaet, får vi at vi må ha en throw ratio på 0.83 : 1. Da er vi inne på en kategori av prosjektorer som kalles short-throw. Noen av disse projektorene klarer produsere bilde på bare noen cm avstand, så 2 meter er bare et tenkt eksempel. Slike projektorer er imidlertid uforholdsmessig dyre, og gir ikke den ønskede effekten på et 3D landskap, siden sollys er svært parallelt lys.

I stedet har vi (altså ChatGPT) funnet fram til noen eksempler på projektorer som kan løse oppgaven på 2 meters avstand:

BenQ TH671ST (short-throw)

- Kastforhold: 0.69~0.83 : 1
- Klarer 100bilde på 1,5 meter
- Perfekt til 240 cm bredde på 2 meter
- 3000 lumen, lav input lag, 1080p

ViewSonic PX706HD

- Kastforhold: 0.61:1
- Gir 240 cm bilde på bare 1,46 meter
- Meget fleksibel på 2 meter
- 3000 lumen, gaming-modus, 1080p

Optoma GT1080HDR

- Kastforhold: 0.5:1
- Gir 240 cm bilde på bare 1,2 meter
- Lett å plassere på 2 m og få 240 cm bilde
- 3800 lumen, 120Hz, HDR

Med priser på 10-15 tusen, er det opp til prosjektet om dette er noe vi skal gå videre med.

Gjenstår:

- Lyder av biltrafikk, folk som vrirler
- Korrekt 3d-modell etter faktisk landskap, med hus og (store) trær, slik at vi får korrekte skygger. Må avventes til modellen er bygget.
- Montere projektor høyt over dioramaet. Vi har et skjermstativ på labben for testing. Vi har skaffet intet mindre enn tre projektorer for test. Disse er åpenbart for små for full-skala, men ok for testing.
- Fargenyanser etter solopp/ned, årstid (kaldt om vinteren, varmt om sommeren).

7 Landskap

7.1 Planleggingsfase

7.1.1 Byggeplate

Vi har basert dioramaet på en kryssfinerplate på 240x120cm. Vi har stivet av med lekter rundt hele, samt montert 8-toms tverrbjelker. Denne platen er montert på understellet fra en gammel heve/senke-pult. Det viste seg svært nyttig å kunne heve/senke platen, spesielt med tanke på arbeidet med elektronikken under bordet.

7.2 Utførelse

7.2.1 Landskap

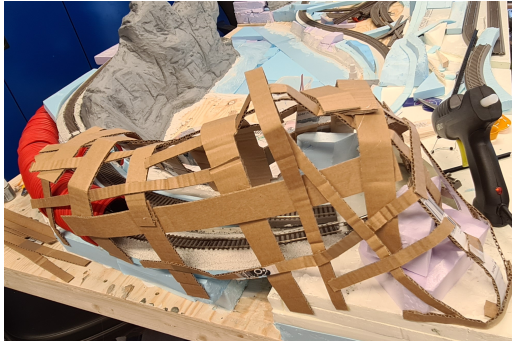
- Landskap formes med isoporplater som legges lagvis og formes med kniv og varmetråd. Dette gir et lett men samtidig stabilt og solid underlag for jernbane.
- Lim og sparkelmasse benyttes for å utjevne overganger og skape naturlige formasjoner.
- Landskap og fjellformasjoner laget med bølgepapp skåret i skiver og strimler, lagt i flettemønster og limt med limpistol.
- Vi bruker avispapir oppbløtt i utvannet trelim/hobbylim som vi legger over lagvis og lar størkne. Dette gir enkelt naturlige former på landskapet.
- Vi sparkler/maler over i grått, brunt og grønt for å imitere landskap.
- Vi har 3D-printet ulike elementer som mur, brohoder, en gammel gullgruve etc. som males og plasseres i landskapet.
- Vi har kjøpt inn byggesett med ulike hus og andre sett med mennesker, iskremvogn, biler etc.
- Autovern er laget med elektrisk kabel og 3d-printede stolper, som så er limt sammen og malt.
- Trær - vi har kjøpt ferdige trær
- Gressmatte - vi har kjøpt gressmatte på rull
- Gress (mose/busker) - vi har kjøpt gress/mose utformet av farget skumgummi
- Vi har lånt en airbrush vi kan bruke til detaljmaling av landskapet.

7.2.2 Vann

Vi har plassert en innsjø i dioramaet med en bunn av isopor, naturstein og sand, og så lager vi vannet med klar epoksy.

Vi har diskutert og testet hvorvidt det er fornuftig å farge epoksyen for å gjøre vannet mer realistisk. I den forbindelse fant vi en anbefaling om å bruke oljebasert maling. Testen viste at vi må være ekstremt forsiktige med dette; kun noen få dråper var nok til at den ble nesten ugjennomsiktig blått.

En alternativ måte å skape illusjonen av dybde på, er å spraymale bunnen i mørkere blåtoner for dypere vann, og dette skal vi også teste.



(a) Pappremser limes..



(b) ..avispapir..



(c) ...sparkles og males

Figur 19: Konstruksjon av fjell

Etter ny test med mindre mengde blåfarge iblandet epoksyen, samt test med airbrushing av bunn, lander vi på at det beste resultatet blir å blande svært forsiktig mengde blått i epoksyen, tømme et bunnlag, og deretter et nytt lag over med helt klar epoksy.

OBS! Da vi helte epoksy i vannbassenget vårt, brukte vi to forskjellige typer. Disse reagerte kjemisk med hverandre, og utviklet varme! Epoksyen ble steinhard i løpet av noen minutter, hevet seg, sprakk, kokte og vi målte 86°C på det varmeste! Vær svært forsiktig med blanding av epoksy!



Figur 20: Brohode



Figur 21: Forlatt gruve

8 Vedlikehold og drift

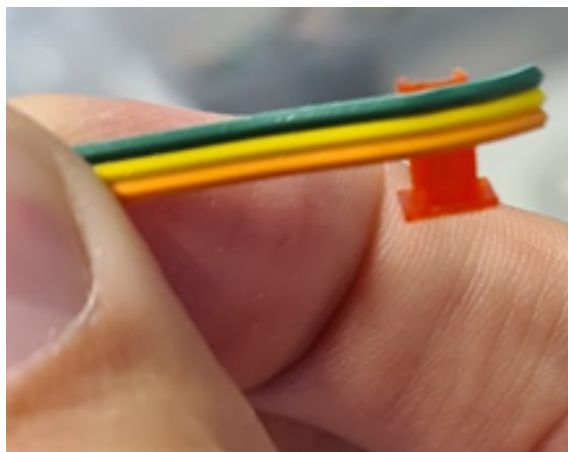
Det viktigste vedlikeholdet som bør gjennomføres jevnlig, er å benytte pusseblokken for å rense skinnene. Lokomotivene og vognene er svært sensitive for støv og andre småpartikler, noe som gjør at togene lett kan spore av dersom sporene er skitne. Det anbefales derfor å rengjøre skinnene før hver gang togsettet tas i bruk. Dersom toget sporer av flere ganger under kjøring, bør man stanse det og rengjøre skinnene på nytt før videre bruk.

Vi anbefaler på det sterkeste at det alltid er en person til stede under kjøring av toget. Som nevnt er togene utsatt for avsporing, og det er viktig at noen er tilgjengelige for å sette toget på sporet igjen ved behov. I tillegg finnes det flere skjøre komponenter i både landskapet og den elektroniske infrastrukturen, som kan ta skade ved uhell.

8.1 Feilsøking

Et godt utgangspunkt for feilsøking er å undersøke MQTT-systemet. Her kan man overvåke dataene som sendes til de ulike topicene, og enkelt identifisere hvilke enheter som eventuelt ikke fungerer som forventet. Dersom dataen i et gitt topic er feil, ligger problemet som oftest hos enheten som publiserer. Er dataen derimot korrekt, men systemet ikke reagerer som det skal, er det mest sannsynlig enheten som abonnerer (subscriber) som har en feil.

De fleste enhetene i systemet kjører små og spesifikke programmer. Et unntak er host-Raspberry Pi-en, som håndterer flere oppgaver og derfor kan være et mer komplekst punkt i systemet ved feilsøking.



Figur 22: Autovern

9 Videreutvikling

Utvikling av en modelljernbane er både en teknisk og kreativ prosess, der mulighetene i praksis er ubegrensede. På grunn av et stramt tidsvindu er flere av komponentene fra ROCO benyttet i dette prosjektet. Det er imidlertid full mulig å utvikle og produsere tilsvarende løsninger selv, slik som gjort på Ringve, der de blant annet har designet eget kretskort for blokkdeteksjon.

Et viktig steg videre vil være å ferdigstille moduler som per i dag ikke er fullført. Dette gjelder blant annet projisering av døgnsyklus på lamndskapet ved hjelp av en projektor, LED-lyslenker som styres i forhold til hurtigklokkens tid og eventuell automatisering av togkjøring. Gjenstående arbeidsoppgaver for de ulike modulene er spesifisert under hver modulbeskrivelse i Kapittel 6.

9.1 Integrering av signallys

Norsk jernbane er i stor grad avhengig av signalanlegg for styring og kontroll av togtrafikken. Systemene er komplekse og krever både inngående teknisk forståelse og omfattende kompetanseinnhenting. For å sikre korrekt implementasjon bør det sterkt vurderes å involvere fagpersoner med erfaring og kompetanse fra jernbanesignalering.

9.2 Andre utviklingsmuligheter

Andre muligheter for videreutvikling inkluderer integrasjon av lyddekoder i ÖBB-lokomotivet for å skape en mer fengende kjøreopplevelse. Samtidig vil det være veldig relevant å utvide selve modelljernbanen. Både anlegget og elektronikken er modulært utviklet og har to spor som føres ut av hovedplaten, noe som åpner for tilkobling av tilleggsplater eller sammenslåing med andre anlegg. Integrasjon av ny sensorikk vil være en relativt enkel prosess, ettersom Raspberry Pi-en i systemet benytter MQTT som kommunikasjonsprotokoll og dermed enkelt kan utveksle data med nye moduler.



Figur 23: Test epoksy uten farge



Figur 24: Test epoksy innblandet blå farge



Figur 25: Bunnen i vannbassenget blir til