

**LAPORAN PRAKTEK KERJA LAPANGAN (PKL)  
FORMULATRIX INDONESIA (PT. PROMANUFACTURE  
INDONESIA)**

**PENGEMBANGAN SIMULATOR PERGERAKAN ROVER ROBOT  
PADA LABORATORIUM BIOTEKNOLOGI**

Diajukan untuk memenuhi sebagian persyaratan Kurikulum Sarjana



Disusun oleh:

Paulina Febrina Siregar	(195150300111010)
Desi Rosda Arum Sari	(195150301111019)
Johannes Riski Sitinjak	(195150301111021)

**PROGRAM STUDI TEKNIK KOMPUTER  
DEPARTEMEN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2022**

## LEMBAR PENGESAHAN

LAPORAN PRAKTEK KERJA LAPANGAN (PKL)  
FORMULATRIX INDONESIA (PT. PROMANUFACTURE INDONESIA)

*PENGEMBANGAN SIMULATOR ROVER ROBOT PADA LABORATORIUM  
BIOTEKNOLOGI*

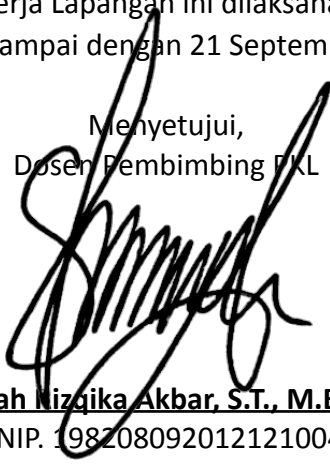
Diajukan untuk memenuhi sebagian persyaratan Kurikulum Sarjana  
Program Studi Teknik Komputer

Disusun Oleh :

Paulina Febrina Siregar	(195150300111010)
Desi Rosda Arum Sari	(195150301111019)
Johannes Riski Sitinjak	(195150301111021)

Praktek Kerja Lapangan ini dilaksanakan pada  
21 Juni sampai dengan 21 September 2022

Menyetujui,  
Dosen Pembimbing PKL



**Sabriansyah Nizqika Akbar, S.T., M.Eng., Ph.D.**

NIP. 198208092012121004

Mengetahui,  
Ketua Departemen Teknik Informatika

**Achmad Basuki, S.T., M.MG, Ph.D**

NIP: 197411182003121002

## PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah laporan Praktek Kerja Lapangan (PKL) ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah laporan Praktek Kerja Lapangan (PKL) ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 November 2022



Paulina Febrina Siregar

195150300111010

## Kata Pengantar

Puji syukur kami ucapkan karena atas berkat rahmat Tuhan Yang Maha Esa kami dapat menyelesaikan laporan Praktek Kerja Lapangan (PKL) ini dengan tepat waktu. Laporan PKL ini berjudul “Pengembangan Pergerakan Rover Robot pada Laboratorium Bioteknologi” yang akan menjelaskan bagaimana proses visualisasi dari *development* hingga *testing*.

Praktek Kerja Lapangan (PKL) merupakan salah satu mata kuliah wajib yang harus ditempuh oleh mahasiswa Teknik Komputer Universitas Brawijaya untuk mendapatkan gelar sarjana. Dengan dilaksanakannya Praktek Kerja Lapangan (PKL) diharapkan mahasiswa dapat menerapkan dan mengimplementasikan ilmu yang telah dipelajari selama masa perkuliahan. Penulisan laporan PKL ini tentunya tidak terlepas dari berbagai pihak terkait yang telah membantu secara langsung maupun tidak langsung. Dalam kesempatan ini kami ingin mengucapkan terimakasih kepada :

1. Bapak Prof. Wayan Firdaus Mahmudy, S.Si., MT., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
2. Bapak Achmad Basuki, S.T, M.MG, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Barlian Henryranu Prasetyo, S.T., M.T., Ph.D. selaku Kepala Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng., Ph.D. selaku dosen pembimbing selama pelaksanaan PKL.
5. Formulatrix Indonesia (PT Promanufacture Indonesia) yang telah memberikan kesempatan untuk melaksanakan PKL.
6. Bapak Ardhian P selaku *leader* project ROVER dan selaku mentor kami selama melaksanakan PKL.
7. Karyawan Formulatrix Indonesia (PT Promanufacture Indonesia) yang tidak dapat kami sebutkan satu persatu.
8. Ayah, Ibu, dan seluruh anggota keluarga yang telah memberi dukungan serta semangat kepada kami.
9. Seluruh pihak yang telah membantu kami selama pelaksanaan PKL dan penulisan laporan PKL yang tidak dapat kami sebutkan satu persatu.

Kami selaku penulisan menyadari bahwa dalam penyusunan serta penulisan laporan Praktek Kerja Lapangan (PKL) ini masih terdapat banyak

kekurangan. Dengan ini kami mengharapkan kritik dan saran yang membangun sebagai bentuk evaluasi untuk kami agar menjadi lebih baik dalam hal penulisan kedepannya. Kami berharap semoga kegiatan PKL dan laporan PKL ini dapat bermanfaat untuk penulis dan pembaca.

Malang, 24 November 2022

Tim Penyusun

Laporan PKL

## Abstrak

Praktek Kerja Lapangan (PKL) dilaksanakan di Formulatrix Indonesia (PT Promanufacture Indonesia), yang dimulai pada tanggal 21 Juni 2022 sampai dengan tanggal 21 September 2022. Formulatrix Indonesia adalah perusahaan yang mengembangkan perangkat keras dan perangkat lunak otomatisasi laboratorium untuk ilmu kehidupan. Salah satu perangkat keras yang diproduksi perusahaan ini yaitu sebuah robot yang dapat memindahkan *plates* dengan otomatis yang bernama Rover. Robot pemindah *plates* atau dalam dunia robotik dikenal dengan *autonomous vehicle*.

Untuk dapat menggunakan robot ini tentunya *user* harus memahami dan mengetahui bagaimana cara kerja, pergerakan robot, dan lintasan yang dapat dilalui oleh robot. Dalam hal ini untuk memudahkan pengguna untuk mengenal robot *autonomous vehicle* (ROVER) dibutuhkan sebuah visualisasi. Visualisasi ini akan menampilkan bentuk animasi dari *autonomous vehicle* (Rover), lintasan yang dapat dilalui, dan pergerakan-pergerakan yang dapat dilakukan *autonomous vehicle* (Rover).

Visualisasi ini berbentuk 3D web, 3D web merupakan perkembangan dari 2D web, pada grafika komputer, 3D merupakan bentuk grafik yang menggunakan representasi data geometri tiga dimensi. Dalam pengembangan visualisasi ini tidak seluruhnya dirancang secara manual, beberapa objek dirancang untuk terhubung otomatis pada JSON data sehingga bentuk objek yang dihasilkan akan berubah sesuai data yang telah dimasukkan. Visualisasi ini dibuat menggunakan *library Three.js* dan dapat dijalankan melalui browser.

Kata kunci : *Autonomous vehicle, 3d visualisasi*

## DAFTAR ISI

<b>LEMBAR PENGESAHAN</b>	<b>ii</b>
<b>PERNYATAAN ORISINALITAS</b>	<b>iii</b>
<b>Kata Pengantar</b>	<b>iv</b>
<b>Abstrak</b>	<b>vi</b>
<b>DAFTAR ISI</b>	<b>vii</b>
<b>DAFTAR GAMBAR</b>	<b>ix</b>
<b>DAFTAR TABEL</b>	<b>xi</b>
<b>BAB 1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang	1
1.2 Rumusan masalah	2
1.3 Tujuan	2
1.4 Manfaat	2
1.5 Batasan Masalah	2
<b>BAB 2 PROFIL INSTANSI</b>	<b>3</b>
2.1 Profil Formulatrix (PT Promanufacture Indonesia Salatiga)	3
2.2 Sejarah Singkat Formulatrix (PT Promanufacture Indonesia Salatiga)	3
2.3 Visi - Misi Formulatrix (PT Promanufacture Indonesia Salatiga)	4
2.3.1 Visi PT Promanufacture Indonesia Salatiga	4
2.3.1 Misi PT Promanufacture Indonesia Salatiga	4
2.4 Nilai - Nilai Perusahaan	4
2.5 Struktur Organisasi Formulatrix (PT Promanufacture Indonesia Salatiga)	5
<b>BAB 3 LANDASAN KEPUSTAKAAN</b>	<b>7</b>
3.1 Tinjauan Pustaka	7
3.2 Dasar Teori	8
3.2.1. Autonomous vehicle	8
3.2.2. Parsing-Data	8

3.2.3 Web Programming Language	9
<b>BAB 4 METODOLOGI</b>	<b>11</b>
4.1 Metodologi	11
4.2 Analisis Kebutuhan	12
<b>BAB 5 HASIL DAN PEMBAHASAN</b>	<b>13</b>
5.1 Perancangan Sistem	13
5.1.1 Informasi komponen-komponen Rover yang dikembangkan dari data JSON/API pada website	13
5.1.2 Pergerakan Rover membawa Plate dari Source ke Destination	19
5.1.3 Pergerakan Rover membawa Plate dari Source ke Destination yang berada di Floor yang berbeda menggunakan Elevator	22
5.1.4 Pergerakan Rover membawa Plate dari Source ke Destination melalui Doors.	23
5.2 Implementasi Website Simulator	24
5.2.1 FM Control Panel Pada Website Simulator	24
5.2.2 Mendefinisikan Visualisasi Instrumen Ke Bentuk 3D	27
5.3 Pengujian Kinerja Website Simulator Rover 3D	38
5.3.1 Tampilan Hasil Akhir Website	38
5.3.2 Side Panel	39
5.3.3 Rover List	40
5.3.5 Object's Name	42
5.3.6 Rover Sedang Menjalankan Task	43
5.3.7 Door Open	45
5.3.8 Elevator Sedang Bergerak	46
<b>BAB 6 PENUTUP</b>	<b>47</b>
6.1 Kesimpulan	47
6.2 Saran	47
<b>DAFTAR PUSTAKA</b>	<b>48</b>
<b>LAMPIRAN DOKUMENTASI KEGIATAN PROGRAM PKL</b>	<b>49</b>



## DAFTAR GAMBAR

Gambar 2.1 Struktur Organisasi Formulatrix	6
Gambar 3.1 Logo Bootstrap	10
Gambar 3.2 Logo Three.JS	10
Gambar 4.1 Diagram Pengembangan Simulator Pergerakan Robot Rover	11
Gambar 5.1 Menampilkan Hasil Parsing-Data pada Inspect Console	14
Gambar 5.2 Data pada Object Power	14
Gambar 5.3 Data pada Edges	15
Gambar 5.4 Data pada vertices	16
Gambar 5.5 Data pada Doors	17
Gambar 5.6 Data pada Elevator	17
Gambar 5.7 Rovers Details	18
Gambar 5.9 Activity Diagram Pergerakan Rover membawa Plate dari Source ke Destination	20
Gambar 5.11 Activity Diagram Pergerakan Rover membawa Plate dari Source ke Destination yang berada di Floor yang berbeda menggunakan Elevator	21
Gambar 5.12 Activity Diagram Pergerakan Rover membawa Plate dari Source ke Destination melalui Doors.	22
Gambar 5.13 FM Control Panel Pada Website Simulator	23
Gambar 5.14 Menu Single Task	23
Gambar 5.15 Menu Multi Task JSON	24
Gambar 5.16 Menu Rover List	25
Gambar 5.17 Bentuk Rover	25
Gambar 5.18 Track Rover	28
Gambar 5.19 Vertices	31
Gambar 5.20 Edges	34
Gambar 5.21 Trip Rover	36
Gambar 5.22 Hasil Website	37
Gambar 5.23 Side Panel	38
Gambar 5.24 Rover List	39
Gambar 5.25 Rover List Details	39

Gambar 5.26 Perspective View	39
Gambar 5.27 Top View	40
Gambar 5.29 Isometric View	41
Gambar 5.30 Object's Name	41
Gambar 5.31 Menampilkan object name pada side-panel	42
Gambar 5.32 Detail object's name	42
Gambar 5.33 Rover Berjalan Menuju Destination	42
Gambar 5.34 Rover Sedang Membawa Plate	43
Gambar 5.35 Door Isopening (sedang membuka)	43
Gambar 5.36 Door Issettled (terbuka)	44
Gambar 5.37 Elevator Bergerak	44

## DAFTAR TABEL

Tabel 3.1 Penelitian Sebelumnya

7

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Praktek Kerja Lapangan (PKL) merupakan kegiatan pembelajaran dan pelatihan yang dilakukan mahasiswa pada dunia kerja untuk menerapkan dan mengimplementasikan ilmu-ilmu yang telah dipelajari selama kegiatan perkuliahan. Dengan adanya pelaksanaan Praktek Kerja Lapangan (PKL) ini diharapkan perguruan tinggi dapat mencetak lulusan-lulusan generasi bangsa yang menguasai ilmu pengetahuan baik secara teoritis maupun aplikatif. Selama masa PKL ini pula diharapkan mahasiswa dapat berproses dan melatih diri untuk beradaptasi di lingkungan kerja yang nyata.

Formulatrix Indonesia (PT. Promanufacture Indonesia) merupakan perusahaan manufaktur yang berfokus pada pengembangan perangkat keras dan perangkat lunak otomatisasi laboratorium untuk ilmu kehidupan. Di perusahaan ini terdapat beberapa produk yaitu diantaranya *Protein Crystallization*, *Liquid Handling*, dan *Filtration*. Dari berbagai macam produk tersebut di dalamnya memiliki produk-produk turunan yang mempunyai berbagai fungsi yang berbeda.

Pada Praktik Kerja Lapangan (PKL) ini kami ditempatkan pada pengembangan produk *Liquid Handling* dengan proyek jenis robot *autonomous vehicle* yang bernama ROVER. Kendaraan yang mampu mengoperasikan dirinya sendiri, tanpa campur tangan manusia secara manual, untuk bergerak menuju tujuan yang telah ditentukan (Pohan dkk, 2019). ROVER digunakan untuk memindahkan *microplate* yang berisi zat protein pada laboratorium yang dapat berjalan pada lintasan yang sudah disediakan dari satu titik ke titik lain (*source* ke *destination*), dimana titik-titik tersebut disebut *vertices*, dan jalur penghubung antar *vertices sources* dan *vertices destination* dinamakan *edges*. *Vertices* merupakan kumpulan *barcode-barcode* yang akan ditangkap oleh robot *Rover* ketika *Rover* sedang berjalan sehingga *Rover* dapat mengetahui arah jalur perjalanannya berdasarkan peta topological yang terdiri dari kumpulan *barcode-barcode vertices* tersebut.

Pada proyek *Rover* ini kami bertugas untuk mengimprovisasi visualisasi website yang akan terintegrasi dengan robot *Rover* guna untuk memonitor pergerakan dari *Rover*. Pada visualisasi yang sedang dikembangkan oleh para developer perusahaan ini, diterapkan pengembangan visualisasi berbasis website 2D dengan menggunakan library *konva.js*. Namun pada proyek ini kami menerapkan pengembangan visualisasi berbasis website 3D dengan tujuan untuk meningkatkan pengalaman pengguna yang lebih *real* dikarenakan visualisasi

objek serta pergerakan yang mirip dengan robot Rover. Adapun website 3D ini sebagian besar dikembangkan menggunakan bahasa pemrograman *javascript* dengan menggunakan *library Three.js*. Agar dapat dieksekusi menggunakan bahasa pemrograman *javascript*, semua data yang dibutuhkan untuk visualisasi seperti data *vertices*, *edges* dan instrumen lainnya yang awalnya bertipe file *.xml* dan *.json*, harus di *parsing* terlebih dahulu menggunakan *javascript*.

## 1.2 Rumusan masalah

Rumusan masalah yang akan menjelaskan mengenai aspek-aspek permasalahan yang akan diselesaikan adalah :

Bagaimana proses pengembangan simulator pergerakan Rover Robot beserta instrumen-instrumennya berdasarkan topologi yang digunakan?

## 1.3 Tujuan

Adapun tujuan dari penulisan laporan Praktek Kerja Lapangan (PKL) ini adalah:

Merangkum seluruh proses pengembangan software simulator pergerakan Rover Robot beserta instrumen-instrumennya berdasarkan topologi yang digunakan.

## 1.4 Manfaat

Manfaat dari pengembangan *website 3D simulator pergerakan Robot Rover* yaitu sebagai berikut :

1. Membantu developer maupun *user* dalam memantau kerja Rover pada laboratorium.
2. Meningkatkan pemahaman user terkait cara kerja Rover dan eksekusi fungsi-fungsi yang ada dengan melakukan simulasi pada website simulator.
3. Meningkatkan pengalaman pengguna (*user experience*) terhadap website lewat improvisasi dari bentuk 2D menjadi 3D.

## 1.5 Batasan Masalah

Batasan masalah untuk proyek Praktek Kerja Lapangan (PKL) ini adalah sebagai berikut:

1. Pengembangan simulator belum mempertimbangkan kecepatan eksekusi sehingga mengakibatkan *lag* atau FPS yang ditampilkan tidak stabil.
2. Lingkungan simulasi hanya bisa mengeksekusi maksimal sebanyak 6 (enam) rover.

## BAB 2 PROFIL INSTANSI

### 2.1 Profil Formulatrix (PT Promanufacture Indonesia Salatiga)

Nama Perusahaan : PT Promanufacture Indonesia (Salatiga)  
Alamat : Jl. Soekarno Hatta No.121, Cebongan, Kec. Argomulyo,  
Kota Salatiga, Jawa Tengah 50731.  
Website : <https://formulatrix.com/>

### 2.2 Sejarah Singkat Formulatrix (PT Promanufacture Indonesia Salatiga)

Formulatrix Indonesia berdiri pada tahun 2002 merupakan perusahaan manufaktur yang berasal dari Amerika Serikat didirikan oleh Jeremy Stevenson. Pusat dari Formulatrix yaitu di Bedford, Massachusetts, Amerika Serikat. Di Indonesia Formulatrix memiliki 4 (empat) cabang yaitu PT Formulatrix Indonesia, PT Promanufacture Indonesia Salatiga, PT Promanufacture Semarang, dan PT Formulatrix Indonesia Bandung. Untuk ketiga cabang merupakan unit produksi sedangkan untuk cabang Bandung hanya berfokus untuk riset. Untuk Formulatrix Amerika Serikat yaitu sebagai pusat yang ditempati oleh tim manajemen puncak dan para *engineer* yang bertugas khusus untuk mengembangkan teknologi baru. Selain itu, Formulatrix juga memiliki cabang di Pakistan yang berfokus pada pengembangan perangkat lunak (*software development center*) dan Cina sebagai kantor penjualan dan pemasaran.

Pada saat masih awal berdiri Formulatrix Indonesia hanya bertempat pada ruko yang beralamat di Jl. Taman Pahlawan Kota Salatiga dan hanya mempekerjakan enam orang. Saat itu Formulatrix hanya akan membuat alat apabila menerima pesanan. Seiring waktu pesanan pun meningkat sehingga membutuhkan tempat lebih luas sehingga pada tahun 2007 Formulatrix memperluas pabrik dengan luas 900m<sup>2</sup> yang beralamat di Tingkir, Kota Salatiga. Sejak saat itu Formulatrix Indonesia mulai mempekerjakan *software engineer*, yang kemudian semakin berkembang mempekerjakan *mechanical engineer* dan *electronic engineer* sehingga terbentuklah tim *Research & Development (R&D)* yang bertugas untuk mengembangkan produk yang telah di riset oleh para *engineer* pusat yang ada di Formulatrix Amerika.

Pada tahun 2006 total jumlah karyawan Formulatrix Amerika Serikat, Indonesia, dan Pakistan yaitu 30 orang dan memiliki sekitar 56 produk terpasang yang terjual ke 32 orang *customer*. Seiring waktu Formulatrix semakin berkembang dan memiliki sekitar 600 lebih karyawan pada tahun 2019, serta bertambahnya produk yang berkembang menjadi sekitar 1837 produk terpasang pada 927 *customer*. Dikarenakan jumlah karyawan dan produk yang bertambah pesat maka kapasitas tempat pun perlu diperluas sehingga dibangunlah PT Promanufacture Indonesia yang berlokasi di Semarang dan Salatiga. PT Promanufacture Indonesia Semarang mulai aktif pada akhir 2018 dan PT Promanufacture Salatiga mulai aktif pada akhir tahun 2019.

Formulatrix memproduksi produk berbasis robotik yang dapat beroperasi dengan otomatis. Produk-produk tersebut diantaranya perangkat lunak yang digunakan untuk mendesain dan menganalisis proses kristalisasi protein yaitu ROCK MAKER®, *liquid handler* yang digunakan sebagai preparasi sampel penelitian dengan produk diantaranya NT8®, FORMULATOR®, MANTIS®, TEMPEST®, F.A.S.T.™, FLO i8™, dan ROVER™. Formulatrix juga memproduksi instrumen yang digunakan untuk proses kristalisasi protein yaitu diantaranya Instrumen untuk proses kristalisasi protein: ROCK IMAGER®, SONIC®, FRAP, MUVIS®, serta memproduksi produk instrumen yang digunakan untuk purifikasi sampel yaitu PULSE – TFF System.

## **2.3 Visi - Misi Formulatrix (PT Promanufacture Indonesia Salatiga)**

### **2.3.1 Visi PT Promanufacture Indonesia Salatiga**

Untuk memungkinkan penemuan ilmiah dalam proteomik dan genomik untuk kemajuan dunia yang bebas penyakit dan bebas kelaparan.

### **2.3.1 Misi PT Promanufacture Indonesia Salatiga**

FORMULATRIX® bekerja sama dengan para peneliti untuk menyederhanakan persiapan dan analisis protein dan asam nukleat dengan merancang solusi tanpa batas dan membawa teknologi mutakhir baru ke industri ilmu hayati. Kami berkomitmen untuk peneliti, laboratorium mereka, dan penemuan ilmiah yang akan meningkatkan kehidupan generasi mendatang.

## **2.4 Nilai - Nilai Perusahaan**

### **1. Intelligence**

Semua masalah diselesaikan paling efisien ketika didekati dengan cara yang bijaksana dan logis. Dalam tantangan pengembangan produk dan pemecahan masalah, kami secara efektif menghilangkan potensi penyebab masalah untuk mendiagnosis kegagalan dengan cepat dan menemukan solusi.

## *2. Hard Work*

Setiap orang menginvestasikan diri mereka di perusahaan melalui 45 jam kerja produktif per minggu.

## *3. Divergent Thinking*

Kami secara mendalam mempertimbangkan solusi dan membawa teknologi dari luar industri untuk menciptakan solusi *out-of-the-box* yang baru. Kami menghargai penemuan pendekatan baru untuk semua aspek perusahaan dan mendorong lingkungan yang dinamis dan kreatif bagi karyawan.

## *4. Sustainability*

Kami sadar biaya dan menyadari dinamika harga pasar untuk memungkinkan perusahaan yang berkelanjutan selama bertahun-tahun yang akan datang.

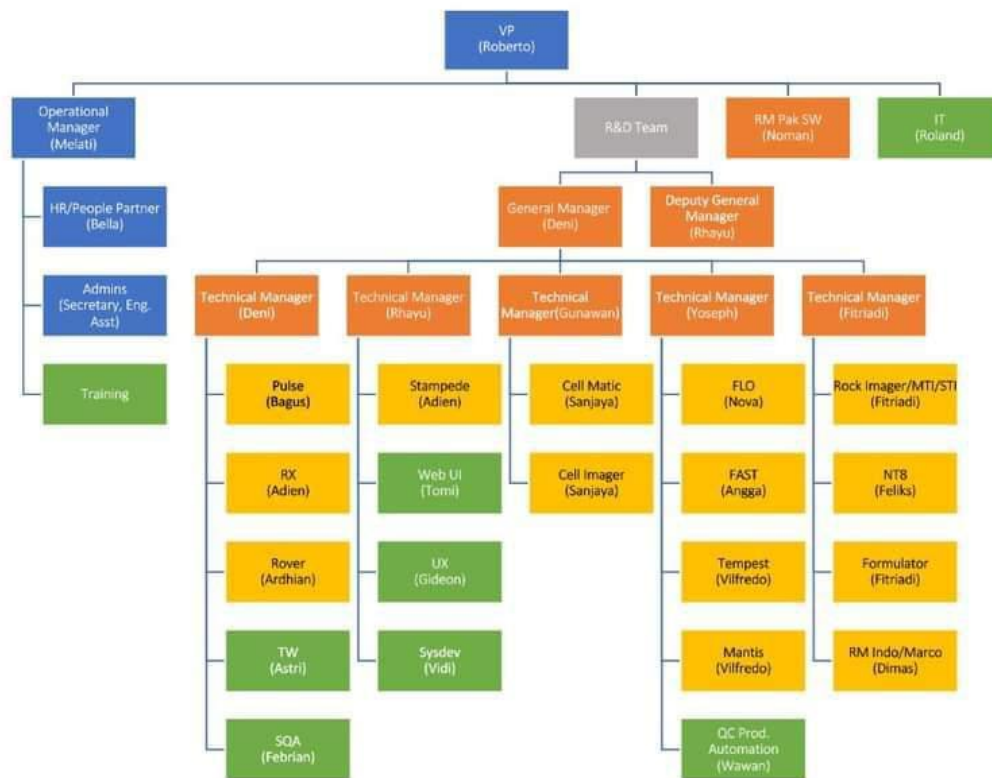
## *5. Integrity*

Tim dan karyawan dapat saling percaya untuk mengembangkan lingkungan kerja yang produktif dan ramah. Kami selalu berterus terang tentang solusi, dan tidak pernah menyesatkan pelanggan mengenai kesesuaian untuk aplikasi mereka.

## **2.5 Struktur Organisasi Formulatrix (PT Promanufacture Indonesia Salatiga)**

Struktur organisasi dari Formulatrix (PT Promanufacture Indonesia Salatiga) terdiri dari *vice president* sebagai pemimpin perusahaan. Operational manager yang membawahi HR, *admins*, dan *training*. *R&D team* yang didalamnya terdapat berbagai technical manager yang membawahi berbagai macam proyek yang dipimpin oleh seorang *leader* pada setiap proyeknya. Gambar dari struktur organisasi dapat dilihat pada **Gambar 2.1** dibawah ini.





**Gambar 2.1 Struktur Organisasi Formulatrix**

## BAB 3 LANDASAN KEPUSTAKAAN

### 3.1 Tinjauan Pustaka

Pada sub-bab ini akan dijelaskan hasil kajian pustaka yang telah dilakukan sebagai dasar dalam pelaksanaan praktik kerja lapangan. Penelitian-penelitian terdahulu yang terkait dengan masalah penelitian ini akan secara umum dijelaskan pada sub-bab ini. Detail persamaan dan perbedaan penelitian terdahulu dan penelitian ini dijelaskan pada **tabel 3.1**. Dari hasil kajian daftar pustaka ini diharapkan dapat mendukung penelitian ini agar memberikan hasil terbaru.

**Tabel 3.1 Penelitian Sebelumnya**

No.	Nama Penulis (tahun), Judul Penelitian	Persamaan	Perbedaan	
			Penelitian Terdahulu	Penelitian PKL
1.	(DongBo Huang et al., 2022). <i>IGAOD: An online design framework for interactive genetic algorithms</i>	Menggunakan <i>Library Three.JS</i> .	Menjadikan algoritma genetika sebagai objek yang diteliti sehingga memerlukan proses pengklasifikasi an dalam pengembangan nnya.	Menjadikan pergerakan robot sebagai objek yang diteliti sehingga menggunakan metode <i>topological-mapping</i> dalam mencari titik perpindahannya.
2.	(Nandi A. K. & Pattanaik L. N., 2020). <i>Design and Development of a Web-Based Robotics Simulator</i>	Menggunakan <i>Library Three.JS</i> .	Menggunakan metode kinematika robot dalam memantau pergerakan simulator robot.	Menggunakan metode <i>topological-map ping</i> dalam mencari titik perpindahannya.
		Menerapkan Simulator Robot berbasis <i>website</i> .		

3.	(Henrique Gaspar, 2022). <i>Current State of the Vessel.JS Library: A Web-Based Toolbox for Maritime Simulations</i>	Menerapkan website simulator dalam visualisasi 3D.	Menggunakan <i>library vessel.JS</i> .	Menggunakan <i>library three.JS</i> .
			Menjadikan transportasi maritim sebagai objek penelitian.	Menjadikan pergerakan robot sebagai objek yang diteliti.

## 3.2 Dasar Teori

### 3.2.1. *Autonomous vehicle*

Pohan dkk (2019) mengatakan bahwa *Autonomous vehicle* merupakan kendaraan yang mampu mengoperasikan dirinya sendiri, tanpa campur tangan manusia secara manual, untuk bergerak menuju tujuan yang telah ditentukan. Robot ini menavigasi dirinya sendiri pada tempat-tempat yang telah ditentukan. Robot ini biasanya menggunakan kamera yang menghadap ke bawah dan GPS sebagai penunjuk arah. Pada robot tertentu pada ruangan terbatas dapat juga menggunakan *barcode* pada lantai dasar yang berfungsi sebagai *map* yang robot ini gunakan sebagai penunjuk arah.

### 3.2.2. *Parsing-Data*

*Parsing* data adalah suatu cara dalam proses pengambilan data pada suatu format yang selanjutnya diubah pada bentuk format lain. *Parsing* data merupakan salah satu pekerjaan wajib yang dilakukan pada saat membuat suatu program. Hal tersebut karena beberapa data akan terbentuk dalam satu kesatuan yang suatu saat akan berguna ketika data tersebut dibutuhkan. Dua bentuk format file yang akan di-*parsing* adalah format JSON dan XML

#### a). JSON

JSON yang memiliki kepanjangan *JavaScript Object Notation* adalah suatu format file yang berbasis teks yang digunakan untuk menyimpan data dan pertukaran data antara *server* dan *client*. JSON data banyak digunakan karena sifatnya yang mudah dipahami, ringkas, dan memiliki data yang terstruktur berdasarkan *syntax Javascript*. Selain itu, JSON data begitu populer karena kompatibel untuk digunakan pada berbagai macam bahasa pemrograman dan *library*. Pada proses perancangan 3D web ini diberikan data JSON yang berisi data-data seluruh komponen yang dibutuhkan. Data tersebut dapat digunakan

untuk membuat titik titik pertemuan antara dua garis atau dapat disebut *vertices*. *Vertices* yang dibutuhkan pada perancangan web ini berjumlah ribuan sehingga tidak memungkinkan untuk dibuat secara manual, maka dari itu dibutuhkan data JSON yang dapat di-*parsing* pada web. Selain itu pada data JSON ini terdapat juga data yang digunakan untuk membuat objek-objek lain pada perancangan 3D web.

#### **b). XML**

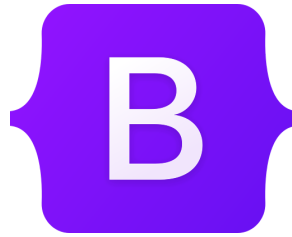
XML yang memiliki kepanjangan *Extensible Markup Language* merupakan bahasa komputer yang berfungsi untuk menyederhanakan dalam proses penyimpanan dan pertukaran data. XML dapat menyimpan data dalam bentuk format teks yang sederhana sehingga data tersebut dapat dengan mudah terbaca oleh server tanpa perlu melakukan perubahan atau modifikasi apapun. Dengan adanya hal tersebut dapat mempermudah pengiriman dan penyimpanan data antar *server*. Pada proses perancangan 3D web ini diberikan data dalam format xml yang digunakan untuk membuat bentuk lintasan robot. Data xml ini merupakan sebuah *global map* yang di-*parsing* pada 3D web sehingga menghasilkan bentuk lintasan yang digunakan sebagai arah robot untuk berjalan, robot hanya bisa berjalan diatas lintasan ini dan tidak bisa keluar dari lintasan.

### **3.2.3 Web Programming Language**

Seperti hakikatnya mengembangkan sebuah website, bahasa pemrograman yang digunakan adalah HTML, CSS dan *Javascript*. Dalam membuat website 3D simulator pergerakan robot ini, kami menggunakan 2 framework yaitu *Bootstrap* dan *Three.JS*.

#### **a) Bootstrap**

*Bootstrap* adalah salah satu toolkit *fronted* yang kuat dan memiliki banyak fitur. *Bootstrap* merupakan sebuah *framework web development* yang berbasis HTML, CSS, dan JavaScript yang dirancang dengan tujuan membuat website yang lebih *responsive*. *Bootstrap* merupakan *framework* gratis yang memiliki sifat *open source*. Selain itu, *bootstrap* menyediakan *script* dan *syntax* yang dapat diterapkan pada berbagai komponen untuk desain web. Pada perancangan 3D web ini *bootstrap* digunakan untuk membangun *side-panel* agar web lebih *responsive* dan *user-friendly*.



Gambar 3.1 Logo Bootstrap

### b) Three.js

*Three.js* adalah *library Javascript* yang banyak digunakan untuk membuat *game* dan 3D visual. *Three.js* dapat dijalankan menggunakan WebGL (*Web Graphic Library*) yang merupakan *API Javascript* yang digunakan untuk melakukan rendering 2D atau 3D grafik sehingga tidak perlu memerlukan *plugins browser* tambahan. *Three.js* memiliki berbagai macam fitur yang kompatibel sehingga sangat cocok digunakan dalam proses membuat *game* ataupun 3D web. Three. Pada *three.js* terdapat beberapa hal yang mendasari dalam proses pembuatan objek yaitu *scene* yang berfungsi untuk membuat objek, *camera* yang berfungsi untuk melihat dari sisi mana objek akan dilihat atau ditampilkan, dan *renderer* digunakan untuk menampilkan objek pada web. Selain itu terdapat *geometry* diantaranya *plane*, *cube*, *box*, *circle* dan masih banyak lagi. Pada perancangan 3D web ini digunakan *planeGeometry* yang di-generated dengan data JSON untuk membuat lintasan. *BoxGeometry* digunakan untuk membuat badan rover dan *cylinderGeometry* untuk membuat kepala rover. Untuk *vertices* berbentuk titik-titik lingkaran sehingga digunakan *circleGeometry* dalam pembuatan objek. Terdapat juga *door* yang dibuat menggunakan *BoxGeometry*. Pada objek elevator dibuat menggunakan *planeGeometry*. Untuk menambahkan warna pada objek-objek tersebut digunakan perintah bernama *MeshMaterial*.

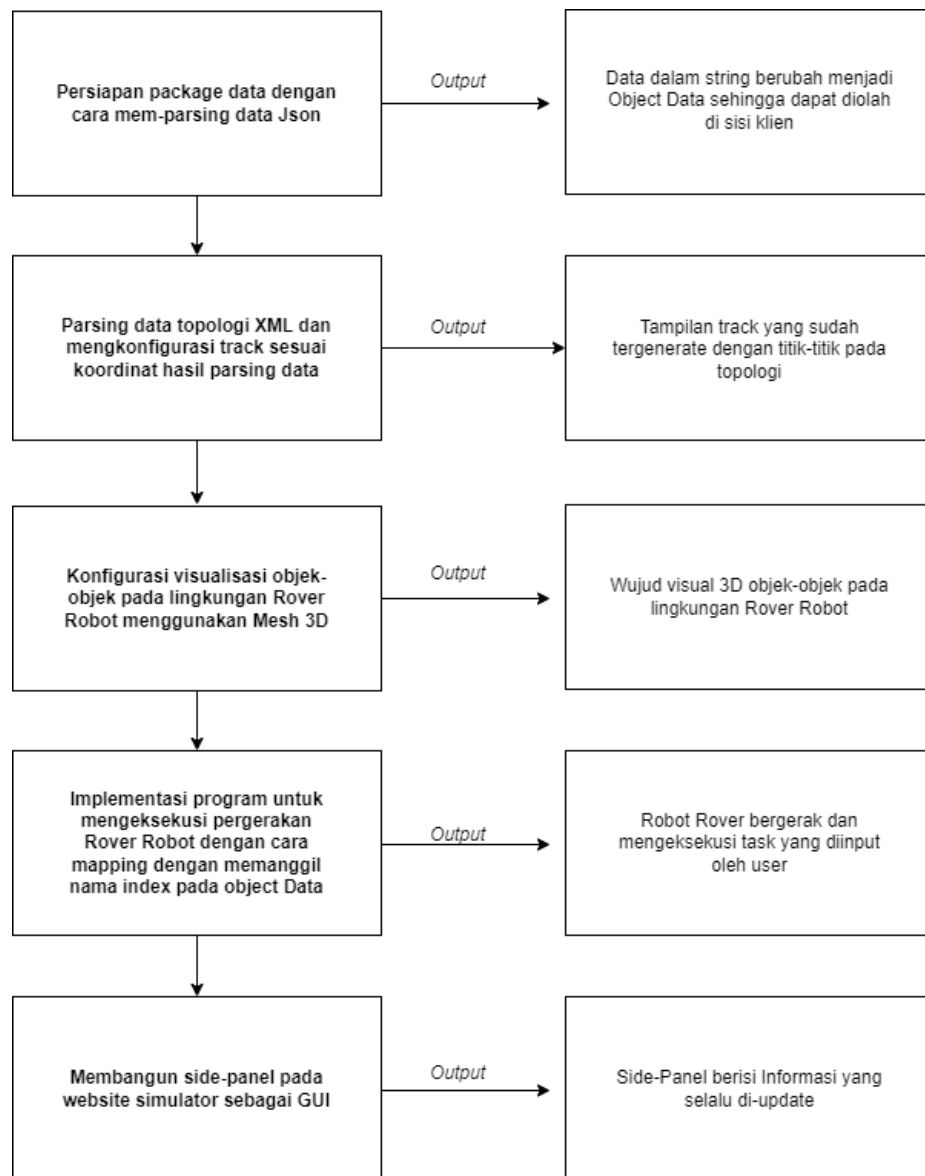


Gambar 3.2 Logo Three.JS

## BAB 4 METODOLOGI

### 4.1 Metodologi

Dibawah ini pada **Gambar 4.1** ditunjukkan sebuah diagram yang menjelaskan metodologi dalam proses pengembangan *website simulator*. Pada diagram tersebut secara berurutan dijelaskan dari tahap awal hingga tahap akhir dalam proses pengembangan.



**Gambar 4.1** Diagram Pengembangan Simulator Pergerakan Robot Rover

Dalam proses pengembangan *website simulator*, adapun pada tahap perancangan awal perlu disiapkan data yang nantinya akan dikembangkan pada sisi *client* untuk membangun *website*. Namun pada saat kita melakukan

pertukaran data JSON pada *web server*, data yang diterima berbentuk string, maka perlu dilakukan *parsing* data menjadi *data object* atau array asosiatif menggunakan *javascript* agar dapat dikembangkan pada sisi *client*. *Output* pada proses ini adalah data yang sebelumnya dalam string berubah menjadi *object data* dan bisa dilihat pada *console web*.

Proses selanjutnya adalah *parsing* data XML yang merupakan topologi dari Robot Rover. Data ini berisi ribuan koordinat yang akan menjadi acuan pengkonfigurasi *track rover* dengan menggambarkan mesh pada setiap koordinat. *Output* dari proses ini adalah visualisasi *track* yang sudah *ter-generated* dengan titik-titik pada topologi.

Tahap ketiga adalah memprogram dan mengembangkan *object data* agar robot rover dapat bergerak dan mengeksekusi task yang diinputkan oleh user. *Output*-nya adalah pergerakan robot rover dalam mengeksekusi task.

Tahapan yang terakhir adalah membangun *side panel* menggunakan *library bootstrap* untuk memudahkan *user* dalam kendali dan pemantauan. *Output* dari proses ini adalah grafik *side-panel* yang dicustom dengan DOM *Javascript* agar *side panel* dapat menampilkan informasi update sesuai dengan pengeksesian Rover.

## 4.2 Analisis Kebutuhan

- 1) Rover :
  1. Rover dapat bergerak membawa *plate* dari *source* ke *destination*.
  2. Rover dapat bergerak membawa *plate* dari *source* ke *destination* yang berbeda lantai menggunakan elevator.
  3. Rover dapat bergerak membawa *plate* melewati *doors*.
- 2) Plate indicator :
  1. *Plate indicator* pada rover menyala berwarna hijau pada saat sedang membawa *plate*.
  2. *Plate indicator* tidak menyala saat rover tidak membawa *plate*.
- 3) Elevator :
  1. Elevator dapat bergerak dari satu lantai ke lantai lainnya.
- 4) Door :
  1. *Door* tertutup saat tidak dilewati rover.
  2. *Door* terbuka saat akan dilewati rover.
- 5) Topologi :
  1. Topologi dari data xml.
  2. Topologi dapat diubah sesuai dengan kebutuhan.
- 6) Pada web terdapat *side panel* yang berisi informasi mengenai rover yang sedang berjalan.

## BAB 5 HASIL DAN PEMBAHASAN

### 5.1 Perancangan Sistem

Perancangan sistem adalah tahapan untuk memberikan gambaran secara rinci bagaimana Rover bisa dikembangkan dan dijalankan. Tahap perancangan sistem yang kami lakukan meliputi mempersiapkan data-data yang berisi informasi sehingga *file* berisi data sudah siap digunakan untuk pengembangan simulator robot *rover* hanya dengan pemanggilan *index*-nya saja. Tahap berikutnya adalah mendefinisikan alur proses aktivitas user dan juga sistem pada saat pengeksekusi *rover* dalam sebuah *activity diagram*.

#### 5.1.1 Informasi komponen-komponen Rover yang dikembangkan dari data JSON/API pada website

Sebelum kami memulai tahap *development* program, penulis terlebih dahulu mengembangkan data yang sudah di-*develop* oleh *backend* dalam bentuk *package JSON data*. Terlebih dahulu agar data antara *browser* dan *server* dapat saling bertukar, maka kita perlu membuka koneksi *websocket* dengan cara membuat *new WebSocket* menggunakan *protocol ws localhost*.

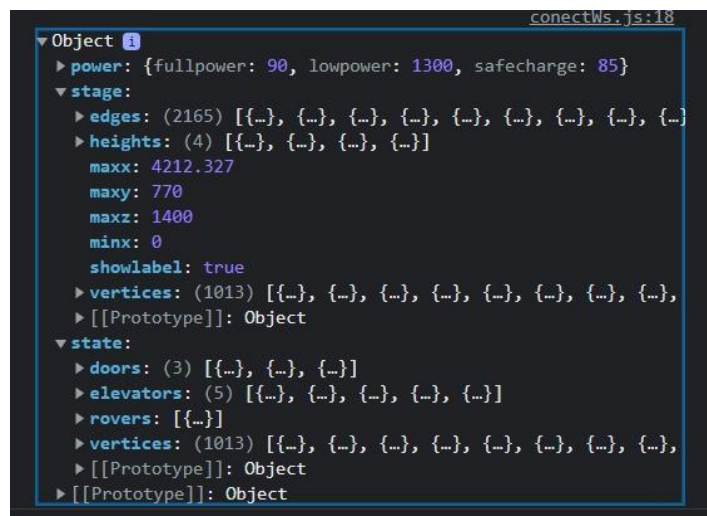
Setelah itu dilakukan parsing data, terdapat 2 jenis JSON data, yaitu: *jsonFullData* dan *jsonEventData*. Perbedaan keduanya adalah untuk *jsonFullData* data yang hanya sekali di parsing sedangkan *jsonEventData* adalah data yang di *parsing* secara terus menerus tiap dalam satu detik, dengan kata lain *jsonEventData* adalah data yang terupdate tiap satu detik.

```
1  var jsonFullData;
2  var jsonEventData;
3
4  function connectWs() {
5      let websocketAddr = "ws://localhost:5000/ws";
6      socket = new WebSocket( websocketAddr);
7      socket.onopen = function (event) {
8          socket.send('full');
9          socket.send('current');
10     };
11     socket.onclose = function (event) {
12     };
13
14     socket.onmessage = function (event) {
15         if (!jsonFullData) {
16             jsonFullData = JSON.parse(event.data);
17             getJsonFullData(jsonFullData);
18             console.log(jsonFullData);
19         }
20         else {
21             jsonEventData = JSON.parse(event.data);
```



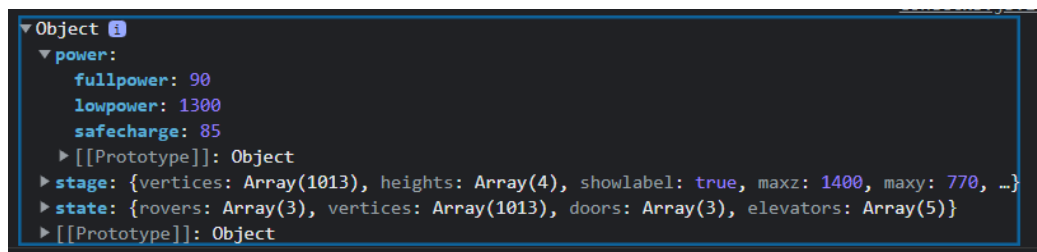
22	<code>getJsonEventData(jsonEventData);</code>
23	<code>getJsonToPanel(jsonEventData);</code>
24	<code>}</code>
25	<code>};</code>
26	<code>};</code>

Setelah membuat program di atas, untuk melanjutkan pengembangan dengan lebih mudah dan sederhana, informasi tentang rover di dalam *json data* yang sudah di-*parsing* sudah dalam bentuk *object* dan dapat dilihat *pada inspect console*, seperti **Gambar 5.1** berikut:



**Gambar 5.1 Menampilkan Hasil Parsing-Data pada Inspect Console**

Pada Object hasil parsing data, terdapat 3 data yaitu *power*, *stage*, dan *state*. **Power** berisi tentang *fullpower*, *low power*, dan *safecharge*, seperti yang dapat dilihat pada **Gambar 5.2** dibawah ini. Sedangkan **Stage** berisi tentang data yang hanya diupdate sekali saat website visualisasi dijalankan dan *state* berisi tentang data yang diupdate setiap satu detik.

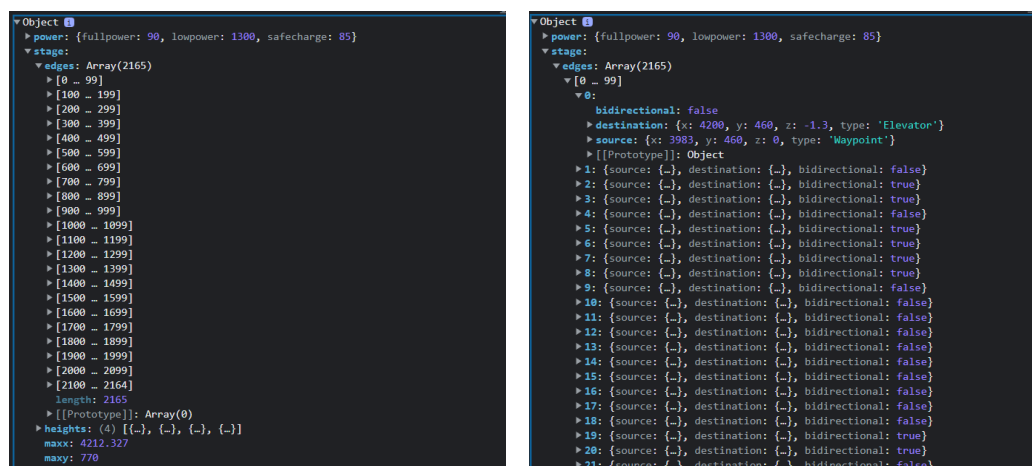


**Gambar 5.2 Data pada Object Power**

Pada **stage** terdapat 3 informasi antara lain: *edges*, *heights*, *maxX*, *maxY*, *min*, dan *vertices*.

- *Edges* merupakan garis penghubung antara *vertex 'source'* dan *vertex 'destination'* sebagai pemberi arah gerak Rover. Data *Edges* disimpan dalam array dengan panjang array sebanyak 2164. Adapun dalam 1 index array menyimpan beberapa data yaitu:
  - *bidirectional* : menentukan apakah edges yang menghubungkan antara satu source ke satu destination memiliki 2 arah, atau tidak. Jika dua arah berarti edges bidirectional, atau nilai bidirectionalnya adalah *'true'*, jika tidak maka nilainya *'false'*.
  - *destination* : *vertices* yang menjadi arah tujuan rover. Berisi data koordinat x, y, z dari *vertices destination* dan *type* dari *vertices*.
  - *source* : *vertices* yang menjadi arah sumber rover. Sama seperti *destination*, *source* berisi data koordinat x, y, z dari *vertices source* dan *type* dari *vertices*.

Pada **Gambar 5.3** berikut terdapat gambar dari object *edges* dan informasi spesifik dalam tiap index array, dan sebagai sampel ditampilkan informasi pada index ke-0.



**Gambar 5.3 Data pada *Edges***

- *Heights* merupakan tingkatan lantai pada topologi yang digunakan.
- *MaxX* merupakan titik X yang paling besar dimana *vertices* berada.
- *MaxY* merupakan titik Y yang paling besar dimana *vertices* berada.
- *Min* merupakan titik koordinat yang paling rendah/pusat yaitu 0,0.
- *Vertices* merupakan titik-titik *node* pada topologi yang berisi berbagai informasi yang dikembangkan agar rover dapat bergerak menjalankan *task*-nya dari satu *vertices* ke *vertices* lain dan melalui *vertices*. *Vertices* pada topologi yang sedang digunakan adalah sebanyak 1013 yang kemudian disimpan dalam satu array. Adapun dalam 1 *index array* berisi beberapa informasi, antara lain:
  - *name* : nama tiap-tiap *vertices*

- Pada **Gambar 5.4** berikut merupakan isi informasi pada *object vertices*, beserta data-data yang ada di dalamnya.



- *Doors* merupakan pintu yang menghubungkan satu ruang ke ruang lainnya, yang nantinya akan dilalui oleh Rover. Pada topologi yang digunakan terdapat 3 *doors* yang kemudian akan dijadikan dalam satu array. Dalam 1 index array terdiri lagi beberapa data informasi, yaitu:

- Isi data informasi tentang door dapat dilihat pada console seperti pada **Gambar 5.5** di bawah ini.



sepanjang banyak rover yang digunakan. Tiap index dalam array Rover berisi data sebagai berikut:

- *Battery* : Rover yang ditambahkan selalu di-charge pada *charging-dock* terlebih dahulu. *Battery* berisi informasi banyak *battery* Rover.
- *Color* : Warna tiap- tiap rover setiap pengekseskusion akan selalu berbeda. Sehingga *color* berisi informasi warna dari rover tersebut.
- *Energy* : energy rover
- *Length* : panjang rover
- *Name* : nama rover yang sedang diaplikasikan.
- *Plate* : indikator yang menunjukkan apakah rover sedang membawa plate atau tidak. Jika ya, maka nilai *plate* adalah 'true', jika tidak maka nilai *plate* adalah 'false'.
- *Spatula* : indikator yang menunjukkan apakah rover sedang melakukan aksi spatula.
- *Theta* : derajat perputaran kepala rover.
- *Trip* : berisi informasi perjalanan rover saat rover sudah mulai berjalan/ dieksekusi.
- *Width* : lebar rover.
- *x* : posisi mula-mula rover pada koordinat x.
- *y* : posisi mula-mula rover pada koordinat y.
- *z* : posisi mula-mula rover pada koordinat z.

Isi data informasi tentang rovers dapat dilihat pada console browser seperti pada **Gambar 5.7** di bawah ini

```
▼ state:
  ▶ doors: (3) [(-), (-), (-)]
  ▶ elevators: (5) [(-), (-), (-), (-), (-)]
  ▼ rovers: Array(3)
    ▶ 0: {name: 'Elva', x: 2669.6963363268806, y: 538.1125645055727, z: 975, width: 105, ...}
    ▼ 1:
      battery: 100
      color: "#CD2EFC"
      energy: 2699
      length: 186
      name: "Dodge"
      plate: false
      spatula: 0
      theta: 89.15580726814491
    ▶ trip: []
      width: 105
      x: 2430.179242209143
      y: 537.8869582549048
      z: 745.57841
    ▶ [[Prototype]]: Object
    ▶ 2: {name: 'Cobra', x: 2428.1808266703824, y: 538.7705441286281, z: 975, width: 105, ...}
      length: 3
    ▶ [[Prototype]]: Array(0)
  ▶ vertices: (1013) [(-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), ...]
  ▶ [[Prototype]]: Object
  ▶ [[Prototype]]: Object
```

**Gambar 5.7 Rovers Details**

- *Vertices* merupakan data vertex yang di-update tiap detik dimana vertices berupa titik-titik topologi pembentuk track rover dan acuan informasi dalam pengembangan rover. *Vertices* merupakan array sepanjang 1013, dimana panjang array tersebut merupakan banyak *vertices* pada topologi yang digunakan. Dalam satu index array terdapat 2 informasi yaitu *name* (nama dari vertex) dan *reserved* (apakah vertex tersebut dipakai dalam pengekseskusion Rover).

Berikut pada **Gambar 5.8** kita bisa melihat dan mengidentifikasi informasi *vertices* pada *object state*.

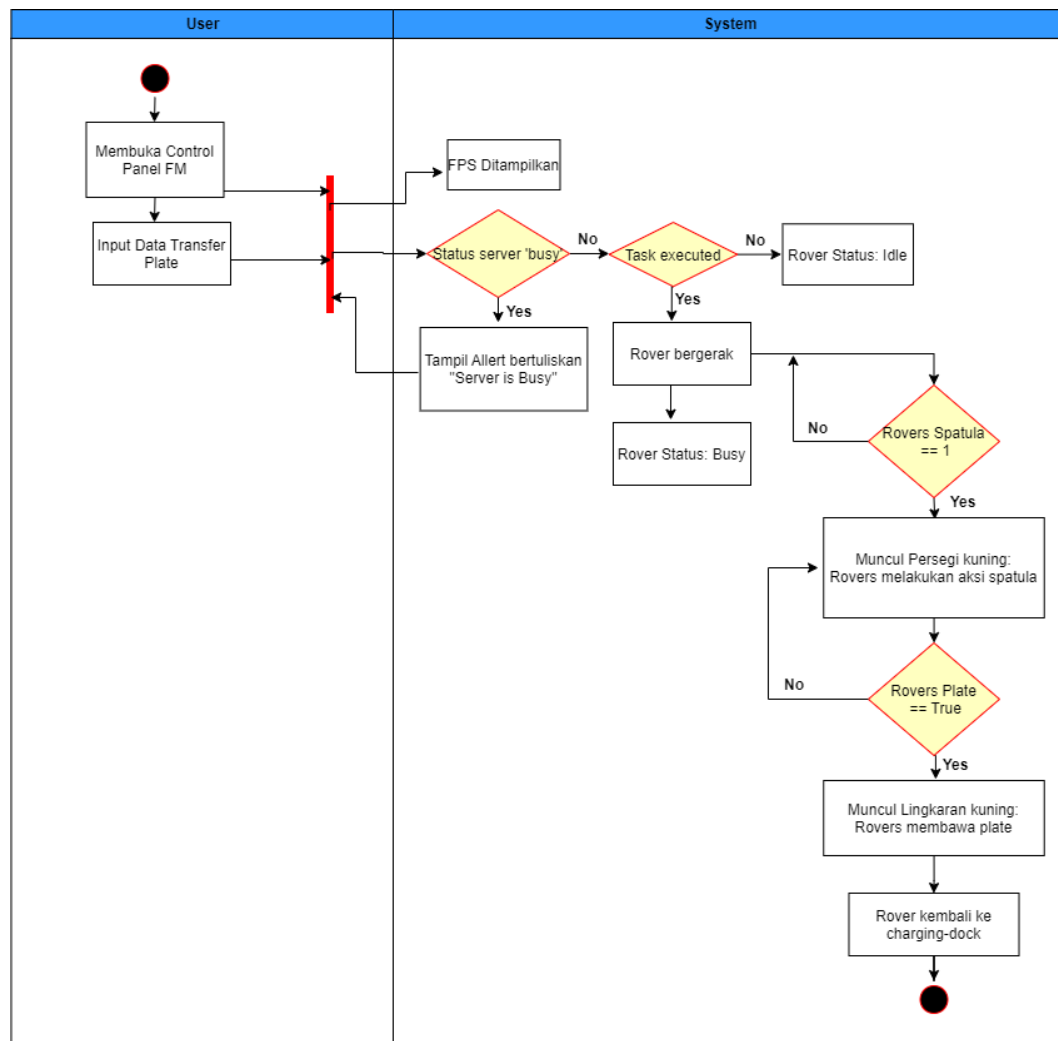
```
▼ state:
  ▶ doors: (3) [(-), (-), (-)]
  ▶ elevators: (5) [(-), (-), (-), (-), (-)]
  ▶ rovers: (3) [(-), (-), (-)]
  ▼ vertices: Array(1013)
    ▼ [0 .. 99]
      ▶ 0: {name: 'delidder-1.l01-01', reserved: ''}
      ▼ 1:
        name: "delidder-1.r01-01"
        reserved: ""
        ▶ [[Prototype]]: Object
      ▶ 2: {name: 'delidder-2.l01-01', reserved: ''}
      ▶ 3: {name: 'delidder-2.r01-01', reserved: ''}
      ▶ 4: {name: 'flo-1.f01-01', reserved: ''}
      ▶ 5: {name: 'flo-1.f02-01', reserved: ''}
      ▶ 6: {name: 'flo-1.f03-01', reserved: ''}
      ▶ 7: {name: 'flo-1.f04-01', reserved: ''}
      ▶ 8: {name: 'flo-1.f05-01', reserved: ''}
      ▶ 9: {name: 'imager-1.i01-01', reserved: ''}
      ▶ 10: {name: 'incubator-1.elevator-left-1', reserved: ''}
      ▶ 11: {name: 'incubator-1.elevator-left-10', reserved: ''}
      ▶ 12: {name: 'incubator-1.elevator-left-11', reserved: ''}
      ▶ 13: {name: 'incubator-1.elevator-left-12', reserved: ''}
      ▶ 14: {name: 'incubator-1.elevator-left-13', reserved: ''}
      ▶ 15: {name: 'incubator-1.elevator-left-14', reserved: ''}
      ▶ 16: {name: 'incubator-1.elevator-left-15', reserved: ''}
      ▶ 17: {name: 'incubator-1.elevator-left-16', reserved: ''}
      ▶ 18: {name: 'incubator-1.elevator-left-17', reserved: ''}
      ▶ 19: {name: 'incubator-1.elevator-left-18', reserved: ''}
      ▶ 20: {name: 'incubator-1.elevator-left-19', reserved: ''}
      ▶ 21: {name: 'incubator-1.elevator-left-2', reserved: ''}
      ▶ 22: {name: 'incubator-1.elevator-left-20', reserved: ''}
```

**Gambar 5.8** Data *vertices* pada *Object State*

### 5.1.2 Pergerakan Rover membawa Plate dari Source ke Destination

Untuk menginstruksikan rover agar dapat membawa *plate* dan bergerak dari *source* ke *destination* yang diinginkan oleh *user*, *user* terlebih dahulu membuka laman *Control Panel FM (Fleet Manager)*, dan memasukkan data *Transfer Plate*. Kemudian sistem akan membuka halaman web visualisasi, yang menampilkan visualisasi rover, informasi pada *side-panel*, dan juga informasi FPS (*Frame per Second*). Apabila status *server* adalah '*busy*' atau sibuk, maka akan muncul tulisan peringatan (*alert*) yang bertuliskan "*Server is Busy*". Namun apabila tidak, maka sistem dapat melanjutkan proses pengkondisian eksekusi *task*. Pada proses pengkondisian, apabila terdapat *task* yang ingin dieksekusi maka Rover akan bergerak menurut *task* yang dimasukkan oleh *user* dan info status Rover pada *side panel* adalah '*busy*'. Namun apabila tidak terdapat *task* yang ingin dieksekusi, maka info status Rover pada *side-panel* adalah '*idle*' atau tidak bekerja dan menunggu hingga menerima *task* dari *user*.

Pada **Gambar 5.9** dibawah ini ditunjukkan *activity diagram* pada saat Rover bergerak membawa *plate* dari *source* ke *destination*, yang dimulai dari user membuka *control-panel FM* hingga selesai.



**Gambar 5.9 Activity Diagram Pergerakan Rover membawa *Plate* dari *Source* ke *Destination***

Saat Rover bergerak mengeksekusi *task*, ia mengkonfirmasi *status spatula action* setiap kali rover melewati *vertices*. Apabila status *spatula action* adalah senilai '1', artinya terjadi *spatula action* atau *spatula* sedang mengambil/meletakkan *plate* pada *vertices* tersebut, pada visualisasi *spatula action* ditandai dengan persegi berwarna kuning di bagian atas rover. Setelah melakukan *spatula action*, status spatula kembali ke nilai '0' yang kemudian dilanjutkan dengan pengkondisian plate indicator. Rover kemudian membawa plate di sepanjang track dan melalui *vertices*, ditandai dengan tampilnya lingkaran berwarna hijau diatas Rover pada visualisasi, sampai pada status *plate indicator* berubah menjadi 'false' dan *spatula action* senilai '1', maka spatula meletakkan *plate* pada *destination*. Jika Rover selesai melakukan *task*-nya, Rover bergerak kembali ke *charging-dock*.

## Potongan Program :

```

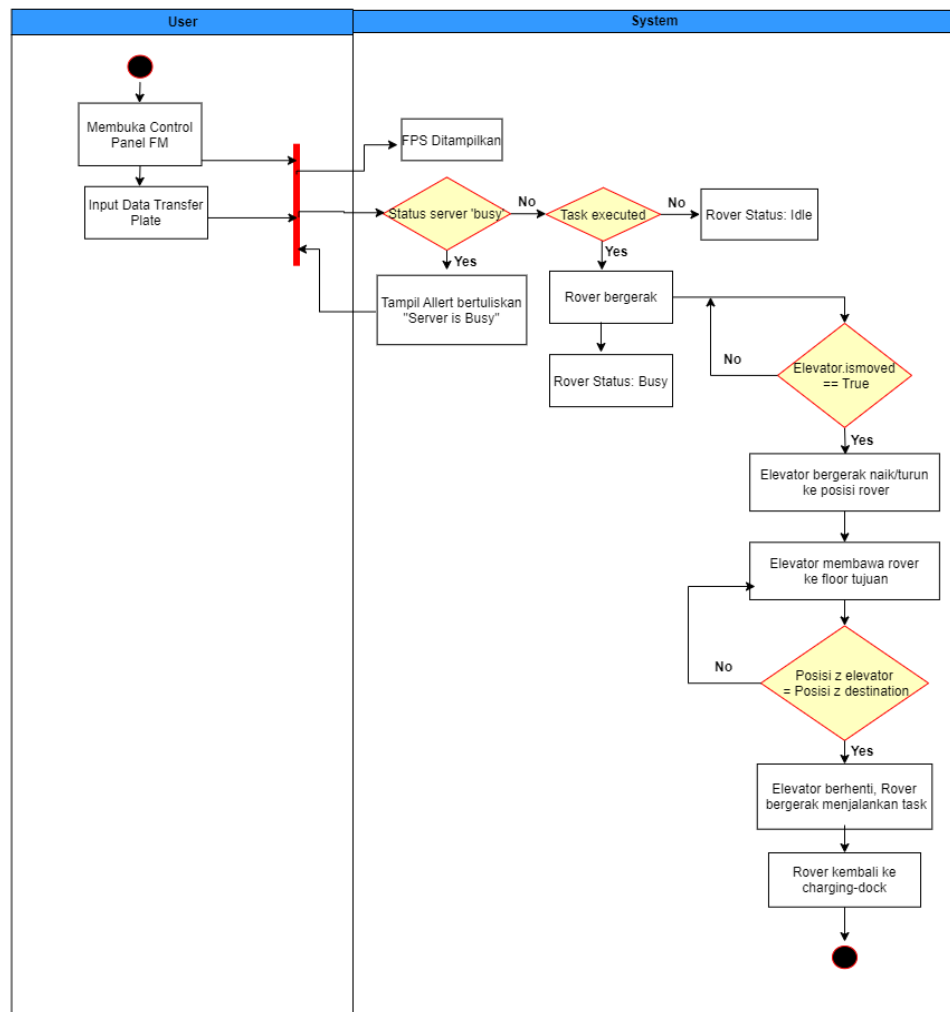
1  //Spatula Action
2      if(myRovers[i].spatula == 1){
3          let mySpatulaGeo= new
4  THREE.PlaneGeometry(myRovers[i].width*0.5/100,
5  myRovers[i].width/100-0.2);
6          let mySpatulaMat = new
7  THREE.MeshBasicMaterial({color: 0xffff00});
8          let spatulaMesh = new THREE.Mesh(mySpatulaGeo
9  mySpatulaMat);
10         spatulaMesh.rotation.x -= Math.PI/2;
11         spatulaMesh.position.y = 0.3;
12         spatulaMesh.position.x = -0.3;
13         myScene.add(spatulaMesh);
14         getRovers[i].add(spatulaMesh);
15     }else if (myRovers[i].spatula ==0){
16         let mySpatulaGeo= new
17  THREE.PlaneGeometry(myRovers[i].width*0.5/100,
18  myRovers[i].width/100-0.2);
19         let mySpatulaMat = new
20  THREE.MeshBasicMaterial({color: myRovers[i].color});
21         let spatulaMesh = new THREE.Mesh( mySpatulaGeo
22  mySpatulaMat);
23         spatulaMesh.rotation.x -= Math.PI/2;
24         spatulaMesh.position.y = 0.3;
25         spatulaMesh.position.x = -0.3;
26         myScene.add(spatulaMesh);
27         getRovers[i].add(spatulaMesh);
28     }
29     //plate indicator
30     if(myRovers[i].plate == true){
31         let plateCly= new
32  THREE.CylinderGeometry(myRovers[i].width*0.5/100-0.25,
33  myRovers[i].width*0.5/100,0.5,50);
34         let plateMat = new
35  THREE.MeshBasicMaterial({color: 0x00ff00});
36         let plateIndicator = new THREE.Mesh( plateCly
37  plateMat);
38         plateIndicator.position.y = 0.01;
39         plateIndicator.position.x = 0.2;
40         getRovers[i].add(plateIndicator);
41     }else if(myRovers[i].plate == false){
42         let plateCly= new
43  THREE.CylinderGeometry(myRovers[i].width*0.5/100-0.25,
44  myRovers[i].width*0.5/100,0.5,50);
45         let plateMat = new
46  THREE.MeshBasicMaterial({color: myRovers[i].color});
47         let plateIndicator = new THREE.Mesh( plateCly
48  plateMat);
49         plateIndicator.position.y = 0.01;
50         plateIndicator.position.x = 0.2;
51         getRovers[i].add(plateIndicator);
52     }
53 }
54

```



### 5.1.3 Pergerakan Rover membawa Plate dari Source ke Destination yang berada di Floor yang berbeda menggunakan Elevator

Pada **Gambar 5.11** dibawah ini ditunjukkan *activity diagram* pada saat Rover bergerak membawa *plate* dari *source* ke *destination* yang berada pada *floor* yang berbeda dengan menggunakan elavator.



**Gambar 5.11 Activity Diagram Pergerakan Rover membawa Plate dari Source ke Destination yang berada di Floor yang berbeda menggunakan Elevator**

Apabila *user* menginstruksikan Rover untuk memindahkan *plate* dari *source* ke *destination* yang berada di lantai berbeda, maka pada saat rover bergerak, program akan selalu meng-*update* status '*elevator.ismoved*'. Apabila status '*elevator.ismoved*' adalah *true*, maka elevator berupa plane akan bergerak naik/turun menuju posisi rover berada. Lalu plane tersebut membawa Rover ke

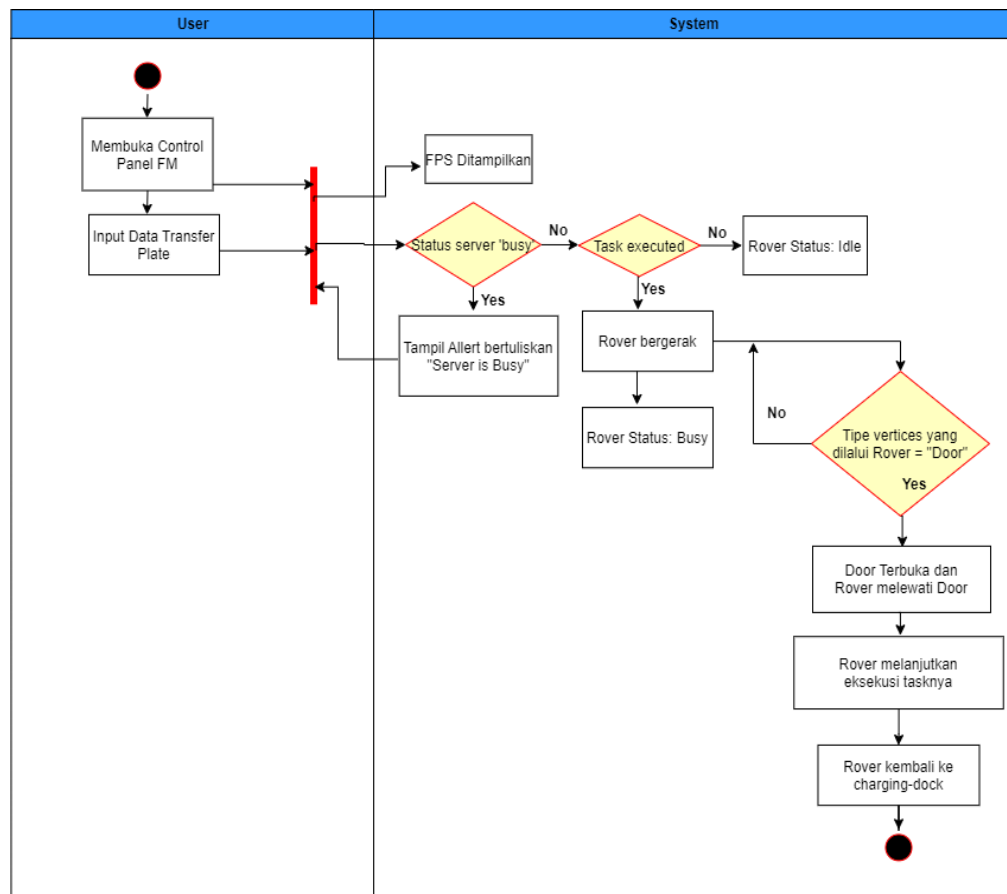
*floor* tujuan, hal ini ditandai dengan posisi z elevator yang sama dengan posisi z *destination*.

Potongan Program :

```
1 //update position elevators
2   for(let i = 0; i < myElevators.length; i++){
3     if(myElevators[i].ismove == true){
4       for(let j = 0; j < getRovers.length; j++){
5         for(let k = 0; k < myRovers[j].trip.length;
6         k++){
7           if(myRovers[j].trip[k].id ==
8           myElevators[i].vertices[0].name || myRovers[j].trip[k].id ==
9           myElevators[i].vertices[1].name
10          ||myRovers[j].trip[k].id ==
11          myElevators[i].vertices[2].name){
12            getRovers[j].position.y =
13            myElevators[i].z/100+0.2;
14          }
15        }
16      }
17    }
18    getElevators[i].position.y =
19    myElevators[i].z/100-0.2;
20  }
21 }
22 }
```

#### 5.1.4 Pergerakan Rover membawa Plate dari Source ke Destination melalui Doors.

Pada **Gambar 5.12** dibawah ini ditunjukkan *activity diagram* pada saat Rover bergerak membawa *plate* dari *source* ke *destination* dengan melalui *doors*.



**Gambar 5.12 Activity Diagram Pergerakan Rover membawa *Plate* dari *Source* ke *Destination* melalui *Doors*.**

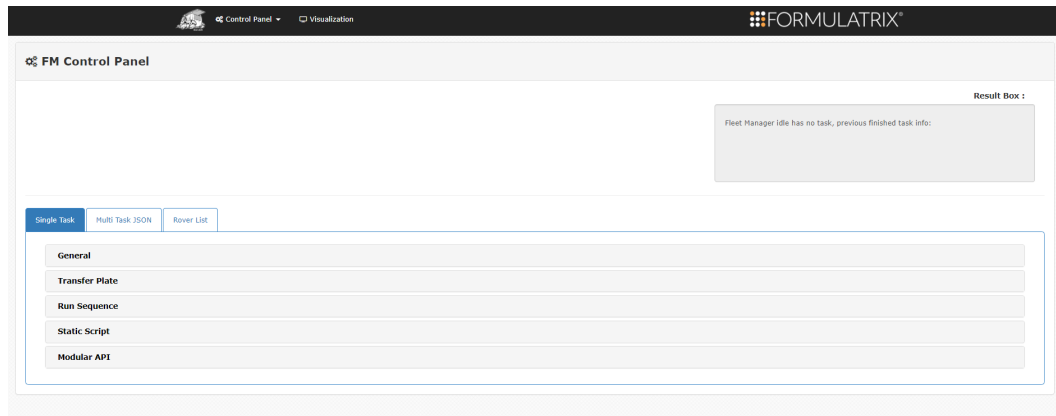
Pada saat Rover bergerak, dan melalui *vertices*, rover akan mengkonfirmasi tipe *vertices* yang dilalui. Apabila tipe *vertices* yang dilalui adalah “*Door*”, maka door berupa plane akan terbuka naik keatas dan rover melaluinya.

## 5.2 Implementasi Website Simulator

Implementasi adalah tahap penerapan dan sekaligus pengujian bagi sistem berdasarkan hasil analisa dan perancangan yang telah dilakukan pada bab sebelumnya. Pada bab ini merupakan implementasi dari hasil pengembangan yang telah dikerjakan.

### 5.2.1 FM Control Panel Pada Website Simulator

Pada **Gambar 5.13** dibawah ini ditunjukkan *FM Control Panel* merupakan kendali untuk menambahkan *task* pada rover.

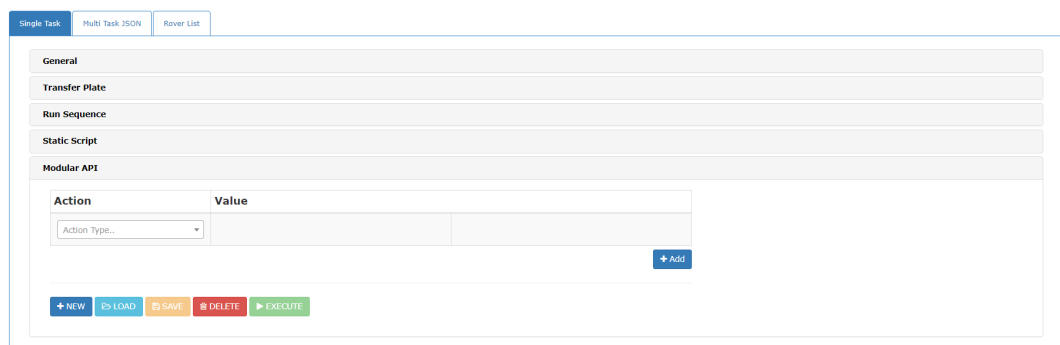


**Gambar 5.13 FM Control Panel Pada Website Simulator**

Pada *FM Control Panel* terdapat beberapa menu dan sub-sub menu.

### 1. Menu Single Task

Pada **Gambar 5.14** dibawah ini ditunjukkan gambar pada menu *single task*.



**Gambar 5.14 Menu Single Task**

Menu *single task* adalah sekumpulan perintah untuk memberikan informasi dan task yang akan diberikan kepada rover untuk dijalankan nantinya oleh rover.

Pada menu *single task*, terdapat beberapa sub menu, seperti

#### a. General

Pada sub menu ini akan memberikan beberapa informasi terkait rover seperti jumlah rover yang aktif, *task* yang sudah selesai dikerjakan oleh rover dan menampilkan visualisasi rover dalam bentuk dua dimensi.

#### b. Transfer Plate

Pada sub menu ini akan memberikan sebuah task kepada satu rover dari titik awal yang akan dituju dan titik akhir yang akan dituju.

#### c. Run Sequence

Pada sub menu ini berfungsi untuk menggerakkan rover ke satu titik yang lebih spesifik berdasarkan titik kordinat yang ditentukan.

#### d. *Static Script*

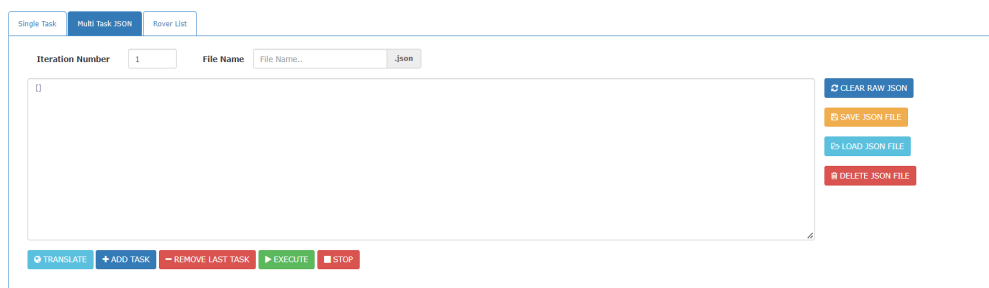
*Static script* berfungsi untuk memberikan komentar pada *task* yang sedang dikerjakan oleh rover.

#### e. Modular API

Pada sub bab ini berfungsi untuk memerintahkan rover mengerjakan task yang sudah didefinisikan sebelumnya.

### 2. Menu Multi Task JSON

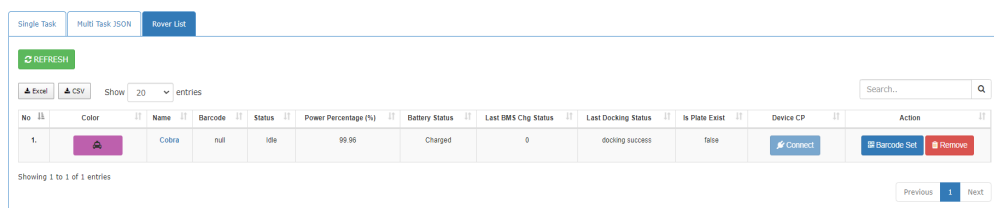
Pada **Gambar 5.15** dibawah ini ditunjukkan gambar pada menu *multi task JSON*.



**Gambar 5.15 Menu Multi Task JSON**

Menu *multi task JSON* merupakan menu untuk memberikan *multi task* kepada rover yang didefinisikan sebelumnya.

### 3. Menu Rover List



**Gambar 5.16 Menu Rover List**

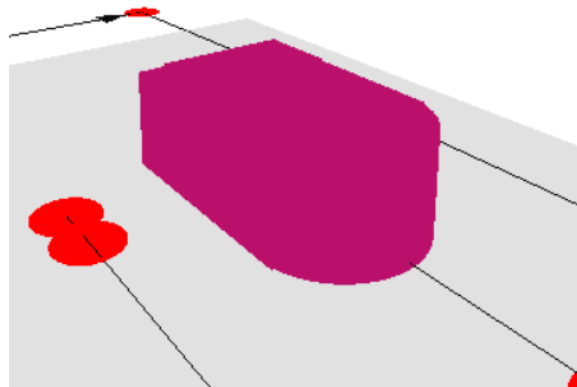
Menu Rover List merupakan menu untuk menampilkan informasi seperti nama, status, *power*, *action* dari rover yang sedang aktif. Menu rover *list* ini dapat dilihat pada **Gambar 5.16**.

### 5.2.2 Mendefinisikan Visualisasi Instrumen Ke Bentuk 3D

Pada bagian ini akan dijelaskan bagaimana semua instrumen yang dibutuhkan divisualisasikan ke bentuk 3D dengan menggunakan *library ThreeJs*. Semua instrumen yang telah didefinisikan ke bentuk 3D nantinya akan digabungkan menjadi satu bagian *website* yang utuh.

#### a. Mendefinisikan Bentuk Rover

Pada **Gambar 5.17** dibawah ini ditunjukkan gambar visualisasi dari bentuk Rover. Hasil akhir dari bentuk 3D Rover adalah penggabungan dari persegi panjang dan setengah tabung, yang dimana setengah tabung berfungsi untuk membentuk bagian dari kepala rover.



**Gambar 5.17** Bentuk Rover

Dibawah ini adalah kode program untuk mendefinisikan rover ke bentuk 3D.

1	//rovers object
2	for (let i = 0; i < myRovers.length; i++) {
3	let myGeometry = new
4	THREE.BoxGeometry(myRovers[i].length / 100 - 0.5,
5	0.5, myRovers[i].width / 100);
6	let myMaterial = new
7	THREE.MeshBasicMaterial({ color: myRovers[i].color
8	});
9	let object = new THREE.Mesh(myGeometry,
10	myMaterial);
11	
12	object.position.x = myRovers[i].x / 100;

13	object.position.y = myRovers[i].z / 100 +
14	0.2;
15	object.position.z = (myRovers[i].y / 100) *
16	-1;
17	
18	//rover head
19	let mycylinder = new
20	THREE.CylinderGeometry((myRovers[i].width * 0.5) /
21	100, (myRovers[i].width * 0.5) / 100, 0.5, 50);
22	let mycynMaterial = new
23	THREE.MeshBasicMaterial({ color: myRovers[i].color
24	});
25	let cylMesh = new THREE.Mesh(mycylinder,
26	mycynMaterial);
27	cylMesh.position.x = 0.55;
28	
29	object.name = myRovers[i].name;
30	cylMesh.name = myRovers[i].name;
31	myScene.add(object);
32	object.add(cylMesh);
33	getRovers.push(object);
34	}

### Penjelasan kode program mendefinisikan rover ke bentuk 3D

	<p>Baris 1-2 Membuat perulangan sebanyak jumlah rover yang sedang aktif.</p> <p>Baris 3-10 Membuat <i>BoxGeometry</i> sebagai bentuk awal rover yaitu bentuk <i>box</i>, setelah mendapatkan bentuk awal, maka diperlukan <i>MeshMaterial</i> yang berfungsi untuk membuat warna pada rover dan yang terakhir yaitu <i>MeshBasicMaterial</i> yang berfungsi untuk menyatukan bagian yang dibuat sebelumnya yaitu <i>Geometry</i> dan <i>Material</i> sebagai bentuk utuh dari rover.</p> <p>Baris 12-16 Menentukan posisi awal x, y, z dari sesuai dengan jumlah rover yang aktif</p> <p>Baris 19-26 Membuat bentuk kepala dari rover yaitu menggunakan <i>CylinderGeometry</i> yang dimana hasilnya akan membentuk sebuah setengah tabung yang akan menjadi kepala dari rover.</p> <p>Baris 27 Menentukan posisi x dari <i>cylinder</i> yang telah dibuat sebelumnya</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>agar membentuk bagian yang utuh dengan badan rover.</p> <p>Baris 29-30 Memberi nama pada <i>object</i> rover sesuai dengan nama-nama yang telah ditentukan sebelumnya.</p> <p>Baris 31 Menambahkan bentuk <i>object</i> rover yang telah dibuat tadi kedalam scene.</p> <p>Baris 32 Mendefinisikan <i>cylinder</i> (kepala) rover sebagai <i>children</i> dari <i>object</i> (badan) dari rover.</p> <p>Baris 33 Memasukkan rover yang sudah utuh kedalam sebuah variabel array yang diberi nama <i>getRovers</i>.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### b. Mendefinisikan Track Rover

*Track Rover* didefinisikan ke bentuk 3D dengan menggunakan *PlaneGeometry* yang ada pada *library ThreeJs*. Pada **Gambar 5.18** dibawah ini ditunjukkan visualisasi dari bentuk *track* Rover.



**Gambar 5.18** *Track* Rover

Dibawah ini adalah kode program untuk mendefinisikan *track rover* ke bentuk 3D.



```

1 fetch("./CCS6_20220726.xml")
2   .then((response) => response.text())
3   .then((data) => {
4       let parser = new DOMParser();
5       var xml = parser.parseFromString(data,
6 "application/xml");
7       var beamMap = xml.getElementsByTagName("Cue");
8       function drawtrack(x, y, z, color) {
9           var trackGeometry = new THREE.PlaneGeometry(0.35,
10 0.35);
11           var trackMaterial = new THREE.MeshBasicMaterial({
12 color: color, side: THREE.DoubleSide });
13           var planeTrack = new THREE.Mesh(trackGeometry,
14 trackMaterial);
15           planeTrack.rotation.x -= Math.PI / 2;
16           planeTrack.position.x = x / 100;
17           planeTrack.position.y = z / 100 - 0.03;
18           planeTrack.position.z = (y / 100) * -1;
19           myScene.add(planeTrack);
20           planeTrack.name = name;
21       }
22
23       let x = [];
24       let y = [];
25       let z = [];
26
27       var xElevator = [(elevator1 = []), (elevator2 = []),
28 (elevator3 = []), (elevator4 = []), (elevator5 = [])];
29       var yElevator = [(elevator1 = []), (elevator2 = []),
30 (elevator3 = []), (elevator4 = []), (elevator5 = [])];
31       for (let i = 0; i < beamMap.length; i++) {
32           x.push(beamMap[i].getAttribute("tx"));
33           y.push(beamMap[i].getAttribute("ty"));
34           z.push(beamMap[i].getAttribute("tz"));
35           if (z[i] != 999999) {
36               drawtrack(x[i], y[i], z[i], 0xeee1e1);
37           } else {
38               switch (beamMap[i].getAttribute("segmentName"))
39 {
40                 //tengah benar
41                 case "IS_ELEVATOR_A_0":
42                     xElevator[0].push(x[i]);
43                     yElevator[0].push(y[i]);
44                     break;
45                 //paling kiri
46                 case "IS_ELEVATOR_B":
47                     xElevator[1].push(x[i]);
48                     yElevator[1].push(y[i]);
49                     break;
50                 //paling kanan benar
51                 case "INC_ELEVATOR01_0":
52                     xElevator[2].push(x[i]);
53                     yElevator[2].push(y[i]);
54                     break;
55                 //ke empat
56                 case "REFRI_ELEVATOR_0":

```

57	<code>xElevator[3].push(x[i]);</code>
58	<code>yElevator[3].push(y[i]);</code>
59	<code>break;</code>
60	<code>//2 dari kiri benar</code>
61	<code>case "CS_ELEVATOR_0":</code>
62	<code>xElevator[4].push(x[i]);</code>
63	<code>yElevator[4].push(y[i]);</code>
64	<code>break;</code>
65	<code>}</code>
66	<code>}</code>
67	<code>}</code>
68	<code>for (var i = 0; i &lt; xElevator.length; i++) {</code>
69	<code>var lengthTrack = (Math.max.apply(Math,</code>
70	<code>xElevator[i]) - Math.min.apply(Math, xElevator[i])) /</code>
71	<code>100;</code>
72	<code>var widthTrack = (Math.max.apply(Math,</code>
73	<code>yElevator[i]) - Math.min.apply(Math, yElevator[i])) /</code>
74	<code>100;</code>
75	
76	<code>var trackGeometry = new</code>
77	<code>THREE.PlaneGeometry(lengthTrack + 0.1, widthTrack +</code>
78	<code>0.1);</code>
79	<code>var trackMaterial = new</code>
80	<code>THREE.MeshBasicMaterial({ color: 0x909090, side:</code>
81	<code>THREE.DoubleSide });</code>
82	<code>var planeTrack = new</code>
83	<code>THREE.Mesh(trackGeometry, trackMaterial);</code>
84	<code>planeTrack.rotation.x -= Math.PI / 2;</code>
85	<code>planeTrack.position.x =</code>
86	<code>Math.min.apply(Math, xElevator[i]) / 100 + lengthTrack</code>
87	<code>/ 2;</code>
88	<code>planeTrack.position.z =</code>
89	<code>(Math.min.apply(Math, yElevator[i]) / 100 + widthTrack</code>
90	<code>/ 2) * -1;</code>
91	<code>getElevators.push(planeTrack);</code>
92	<code>planeTrack.name = "Elevator";</code>
93	<code>myScene.add(planeTrack);</code>
94	<code>}</code>
95	<code>});</code>
96	
97	
98	
99	
100	

### Penjelasan kode program mendefinisikan Track ke bentuk 3D

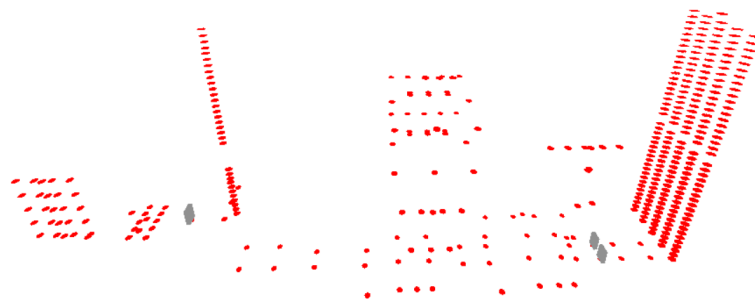
	<p>Baris 1-8 Meng-<i>import</i> sebuah file xml yang kemudian diparsing dan kemudian dimasukkan kedalam sebuah variabel yang bernama <i>beamMap</i>.</p> <p>Baris 9</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Membuat sebuah fungsi yang bernama <i>drawtrack</i> yang memiliki beberapa parameter yaitu x, y, z dan <i>color</i> yang berfungsi untuk menentukan posisi dan warna dari <i>track</i> yang akan dibuat.</p> <p>Baris 10-23</p> <p>Mendefinisikan bentuk dari <i>track</i> yang akan dibuat yaitu dengan menggunakan <i>PlaneGeometry</i>, <i>MeshBasicMaterial</i> dan <i>Mesh</i> yang akan menghasilkan bentuk 3D seperti lantai dengan posisi dan warna yang ditentukan sesuai dengan parameter dari fungsi yang telah dibuat dan setelah semua selesai dibuat kemudian ditambahkan kedalam <i>scene</i>.</p> <p>Baris 25-34</p> <p>Mendeklarasikan variabel x, y, z dengan tipe data array yang berfungsi untuk menampung titik x, y dan z elevator yang ada.</p> <p>Baris 35-74</p> <p>Perulangan untuk melakukan <i>push</i> atau memasukkan nilai x, y dan z ke dalam array yang sudah dideklarasikan sebelumnya.</p> <p>Baris 75</p> <p>Melakukan perulangan atau <i>looping</i> sebanyak jumlah dari elevator yang dimana perulangan ini berfungsi untuk membuat bentuk dari elevator ke bentuk 3D.</p> <p>Baris 76-81</p> <p>Menentukan panjang dan lebar dari elevator dengan cara mengurangi nilai x maksimum dengan nilai x minimal dan mengurangi nilai y maksimal dengan nilai y minimum.</p> <p>Baris 83-90</p> <p>Membuat bentuk elevator kedalam bentuk 3D dengan menggunakan <i>PlaneGeometry</i>, <i>MeshBasicMaterial</i> dan <i>Mesh</i>.</p> <p>Baris 91-97</p> <p>Menentukan arah dan posisi x, y dan z dari elevator.</p> <p>Baris 98-100</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	Memasukkan semua elevator yang ada ke dalam array, selanjutnya memberi nama dari elevator kemudian menambahkan objek 3D elevator yang telah dibuat ke dalam <i>scene</i> .
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### c. Mendefinisikan Vertices

Pada **Gambar 5.19** dibawah ini ditunjukkan gambar dari bentuk visualisasi *vertices*. *Vertices* dibuat menggunakan *CircleGeometry* yang ada pada *library ThreeJs* yang dimana akan menghasilkan lingkaran *flat* yang diberi warna merah.



**Gambar 5.19 Vertices**

Dibawah ini adalah kode program untuk mendefinisikan *vertices* ke bentuk 3D.

1	<code>//vertices</code>
2	<code>for (var i = 0; i &lt; getVertices.length; i++) {</code>
3	<code>var myGeoVertices = new</code>
4	<code>THREE.CircleGeometry(0.2, 20);</code>
5	<code>var myMatVertices = new</code>
6	<code>THREE.MeshBasicMaterial({ color: 0xff0000, side:</code>
7	<code>THREE.DoubleSide });</code>
8	<code>var objectVertices = new</code>
9	<code>THREE.Mesh(myGeoVertices, myMatVertices);</code>
10	
11	<code>objectVertices.rotation.x -= Math.PI / 2;</code>
12	<code>objectVertices.position.x = getVertices[i].x /</code>
13	<code>100;</code>

14	objectVertices.position.y = getVertices[i].z /
15	100;
16	objectVertices.position.z = (getVertices[i].y
17	/ 100) * -1;
18	
19	objectVertices.name = getVertices[i].name;
20	myScene.add(objectVertices);
21	
22	//Doors
23	if (getVertices[i].type == "Door") {
24	var myGeoDoors = new
25	THREE.BoxGeometry(0.25, 0.7, 0.8);
26	var myMatDoors = new
27	THREE.MeshBasicMaterial({ color: 0x909090, side:
28	THREE.DoubleSide });
29	var myMeshDoors = new
30	THREE.Mesh(myGeoDoors, myMatDoors);
31	
32	myMeshDoors.rotation.x -= Math.PI / 2;
33	myMeshDoors.position.x = getVertices[i].x
34	/ 100;
35	myMeshDoors.position.y = getVertices[i].z
36	/ 100 + 0.4;
37	myMeshDoors.position.z = (getVertices[i].y
38	/ 100) * -1;
39	myMeshDoors.name = "Door";
40	myScene.add(myMeshDoors);
41	getDoors.push(myMeshDoors);
42	doorsPosition.push(getVertices[i].z / 100
43	+ 0.4);
44	}
45	}

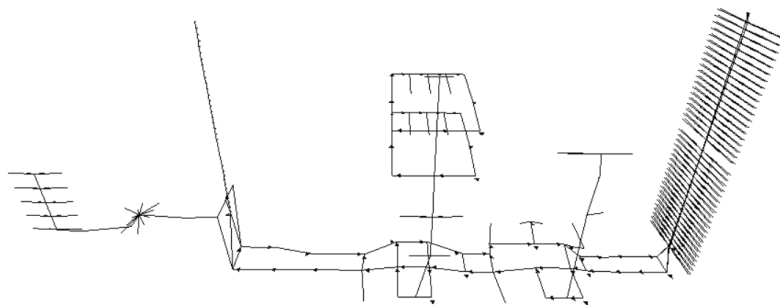
### Penjelasan kode program mendefinisikan Vertices ke bentuk 3D

	<p>Baris 2 Melakukan perulangan sebanyak jumlah dari <i>vertices</i> yang ada.</p> <p>Baris 3-9 Membuat bentuk <i>vertices</i> ke dalam bentuk 3D dengan menggunakan <i>CircleGeometry</i>, <i>MeshBasicMaterial</i> dan <i>Mesh</i> yang dimana nantinya akan menjadi berbentuk lingkaran berwarna merah.</p> <p>Baris 11-17 Menentukan nilai x, y dan z yang akan menjadi posisi dari setiap <i>vertices</i> yang ada.</p> <p>Baris 19-20</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Memberikan nama pada setiap <i>vertices</i> yang ada sesuai dengan yang sudah ada kemudian ditambahkan kedalam <i>scene</i>.</p> <p>Baris 23 Membuat kondisi <i>if</i> yang berfungsi untuk membuat pintu dalam bentuk 3D.</p> <p>Baris 24-30 Membuat bentuk door ke dalam bentuk 3D dengan menggunakan <i>BoxGeometry</i>, <i>MeshBasicMaterial</i> dan <i>Mesh</i> yang akan menghasilkan door berbentuk balok berwarna abu-abu.</p> <p>Baris 32-38 Memberikan nilai <i>x</i>, <i>y</i> dan <i>z</i> dari door untuk menentukan posisi dari door tersebut.</p> <p>Baris 39-43 Memberikan nama pada setiap <i>door</i> kemudian menambahkan <i>door</i> ke dalam <i>scene</i> kemudian di <i>push</i> ke dalam sebuah array yang sudah dideklarasikan sebelumnya agar dapat diupdate posisinya.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### d. Mendefinisikan Edges

*Edges* dibuat ke bentuk 3D menggunakan *LineBasicMaterial* yang ada pada *library ThreeJs* yang dimana akan menghasilkan *edges* berbentuk garis berwarna hitam. Pada **Gambar 5.20** dibawah ini ditunjukkan gambar bentuk visualisasi dari *edges*.



**Gambar 5.20 Edges**

**Dibawah ini adalah kode program untuk mendefinisikan *edges* ke bentuk 3D.**

1	//edges
2	for (let i = 0; i < JsonDataEdges.length - 1; i++)
3	{
4	// if(JsonDataEdges[i].destination.type ==
5	"Waypoint"){
6	let points = [];
7	let lineTrack = new THREE.LineBasicMaterial({
8	color: 0x0a0a0a, linewidth: 0.001 });
9	points.push(new
10	THREE.Vector3(JsonDataEdges[i].source.x / 100,
11	JsonDataEdges[i].source.z / 100,
12	(JsonDataEdges[i].source.y / 100) * -1));
13	points.push(new
14	THREE.Vector3(JsonDataEdges[i].destination.x / 100,
15	JsonDataEdges[i].destination.z / 100,
16	(JsonDataEdges[i].destination.y / 100) * -1));
17	let lineGeo = new
18	THREE.BufferGeometry().setFromPoints(points);
19	let lineEdges = new THREE.Line(lineGeo,
20	lineTrack);
21	myScene.add(lineEdges);
22	points = [];

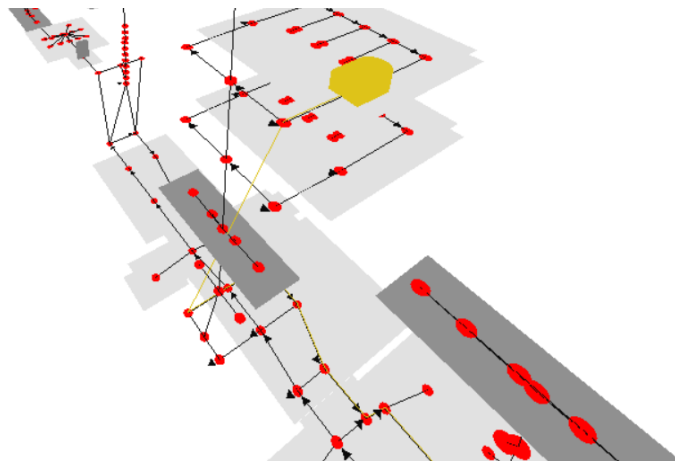
**Penjelasan kode program mendefinisikan *Edges* ke bentuk 3D**

	<p>Baris 2-3 Membuat perulangan for sebanyak jumlah <i>edges</i> yang akan dibuat ke bentuk3D.</p> <p>Baris 4-5 Kondisi if untuk memilih <i>edges</i> yang ingin diubah ke bentuk 3D hanyalah yang bertipe "waypoint".</p> <p>Baris 6-8 Membuat <i>edges</i> ke bentuk 3D dengan menggunakan <i>LineBasicMaterial</i> yang akan menghasilkan bentuk 3D berupa garis.</p> <p>Baris 9-16 Memasukkan vektor ke dalam sebuah array yang nantinya akan digunakan untuk menentukan titik awal dan akhir garis (<i>edges</i>) yang akan dibuat.</p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Baris 17-20 Menambahkan geometry dari edges dengan menggunakan <i>BufferGeometry</i> agar garis (<i>edges</i>) yang dibuat berbentuk 3D.</p> <p>Baris 21-22 Menambahkan <i>edges</i> yang sudah berbentuk 3D ke dalam <i>scene</i>.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### e. Mendefinisikan Trip Rover

Pada **Gambar 5.21** dibawah ini ditunjukkan gambar visualisasi dari bentuk *trip* Rover. Trip Rover divisualisasikan dengan sebuah garis yang warnanya sama dengan warna rover yang sedang dieksekusi. Garis *trip* rover tersebut akan selalu ter-*update* setiap detik seiring rover melaju melaluinya.



**Gambar 5.21 Trip Rover**

<pre> 1 2 3 4 5 6 7 8 9 10 11 12 13 </pre>	<pre> function getTrip(point, roverColor){     let lineTrack = new THREE.LineBasicMaterial({color : roverColor,linewidth : 3})     let lineGeo = new THREE.BufferGeometry().setFromPoints(point)     let lineRovers = new THREE.Line(lineGeo, lineTrack);     lineRovers.name = "roversTrip";     myScene.add(lineRovers);     const removeLine =setTimeout(function(){         let selectedObject = myScene.getObjectByName("roversTrip");         myScene.remove(selectedObject);     },1000); } </pre>
--------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



14	<code>},1000);</code>
15	<code>};</code>
16	

#### Penjelasan kode program mendefinisikan Trip Rover ke bentuk 3D

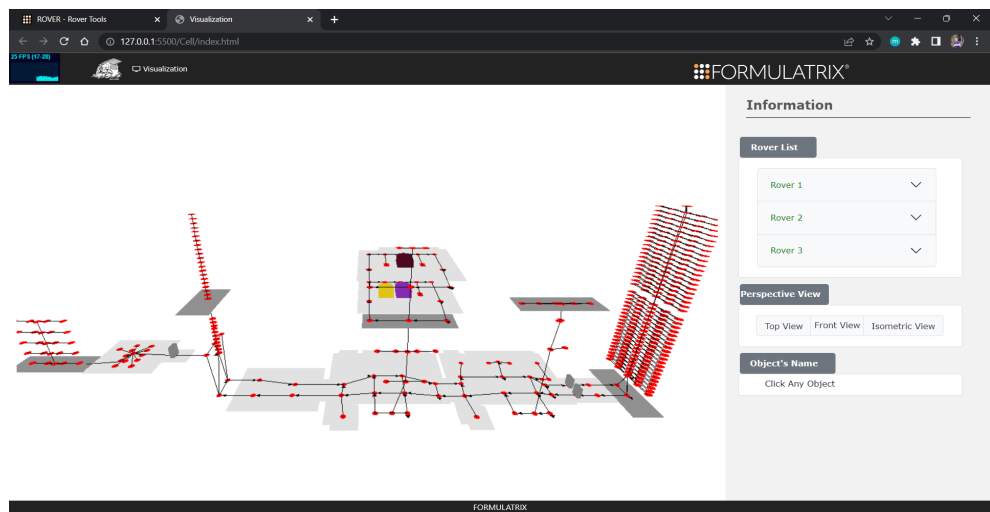
Baris 1	Membuat sebuah fungsi dengan parameter <i>point</i> dan <i>roverColor</i> yang akan diupdate secara terus menerus.
Baris 2-8	Membuat <i>trip</i> rover berupa garis berbentuk 3D dengan menggunakan <i>LineBasicMaterial</i> yang akan memberikan warna sesuai dengan warna rover dan <i>BufferGeometry</i> yang digunakan sebagai geometrinya.
Baris 9-10	Memberikan nama untuk setiap <i>trip</i> dari rover kemudian menambahkannya ke dalam <i>scene</i> .
Baris 11-15	Membuat sebuah variabel sebagai <i>anonymous function</i> yang berfungsi untuk menghapus <i>trip</i> rover yang sudah dilewati dengan membuat <i>setTimeout</i> yang akan di update setiap satu detik.

### 5.3 Pengujian Kinerja Website Simulator Rover 3D

Pada sub bab ini akan dijelaskan bagaimana hasil dari pengujian website Simulator Rover 3D dan bagian-bagian dari website secara mendetail serta fungsi dari setiap bagian website yang telah jadi.

#### 5.3.1 Tampilan Hasil Akhir Website

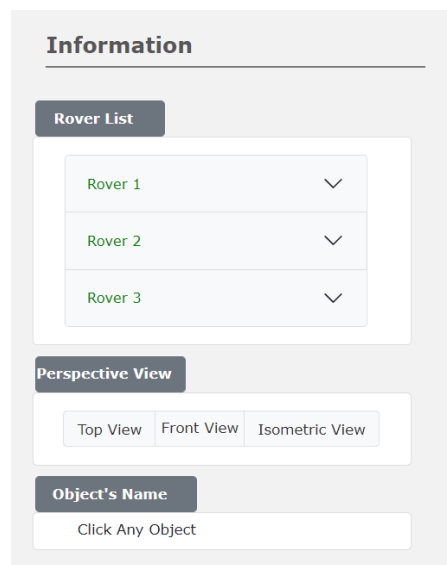
Pada **Gambar 5.22** dibawah ini ditunjukkan gambar dari hasil akhir *website* yang telah dikembangkan.



**Gambar 5.22 Hasil Website**

### 5.3.2 Side Panel

Pada **Gambar 5.23** dibawah ini ditunjukkan gambar dari *side panel*.

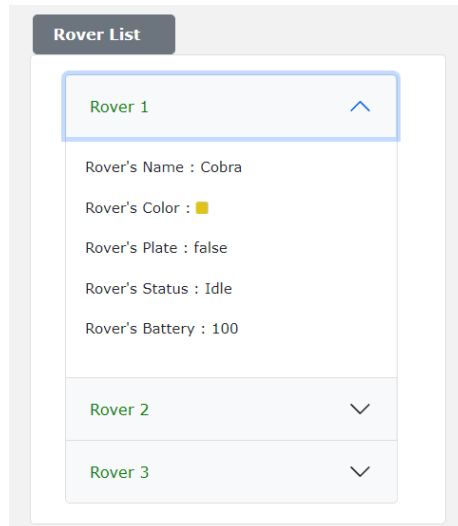


**Gambar 5.23 Side Panel**

Pada *Side Panel* Website terdapat beberapa informasi dan perintah yaitu, informasi dari *Rover List* yang akan memberikan informasi jumlah dari rover yang aktif. Terdapat juga tombol *Perspective View* yaitu tombol untuk mengubah arah sudut pandang dari tampilan simulator yang terdiri dari 3 sudut pandang yaitu *Top View*, *Front View* dan *Isometric View*. Selanjutnya ada *Object's Name* yang akan memberikan informasi nama objek dari simulator jika di klik.

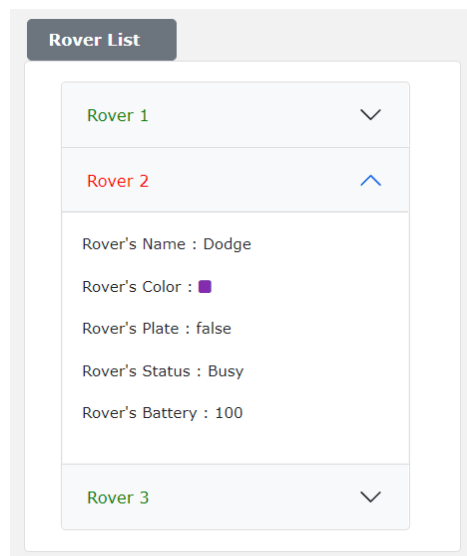
### 5.3.3 Rover List

Pada **Gambar 5.24** dibawah ini ditunjukkan gambar pada menu *Rover List*.



**Gambar 5.24 Rover List**

Pada *Rover List* terdapat informasi yang menunjukkan jumlah dari rover yang aktif, kemudian setiap rover yang aktif terdapat informasi yang lebih spesifik yaitu nama, warna, status, *plate*, baterai.

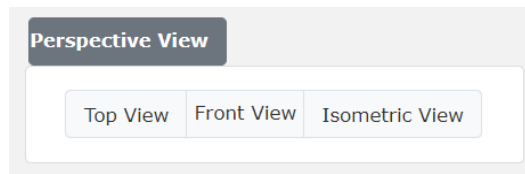


**Gambar 5.25 Rover List Details**

Seperti pada **Gambar 5.25** di atas, jika status rover sedang "*Busy*" maka warna dari tombol rover akan berwarna merah dan jika status rover sedang "*idle*" maka warna tombol rover akan berwarna hijau.

### 5.3.4 Perspective View

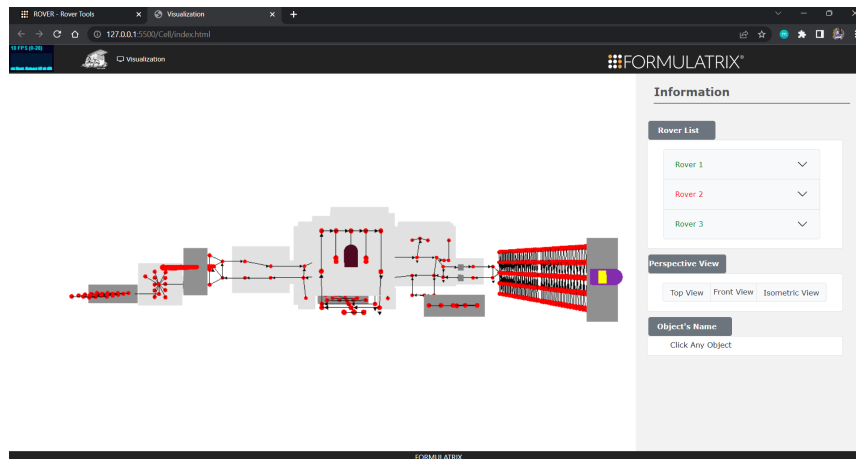
Pada **Gambar 5.26** dibawah ini ditunjukkan gambar dari menu *perspective view*.



**Gambar 5.26** *Perspective View*

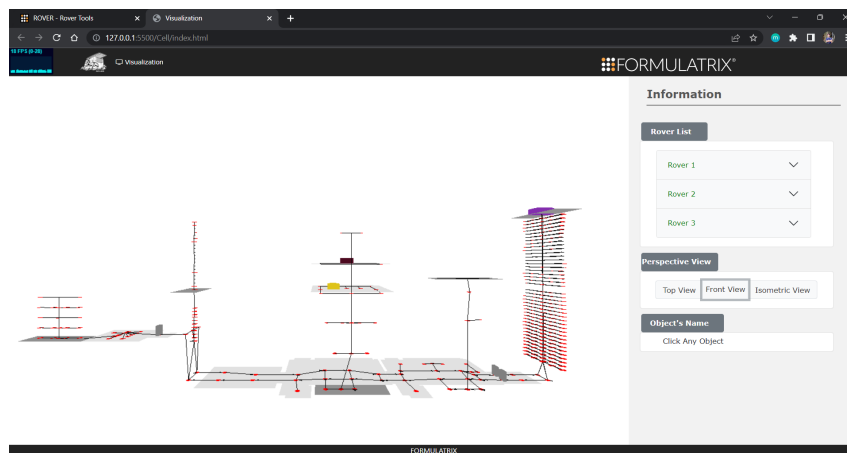
Pada tombol *perspective view* terdapat tiga tombol yang berfungsi untuk mengubah arah sudut pandang visualisasi 3D.

Pada **Gambar 5.27** dibawah ini ditunjukkan gambar dari menu *top view*



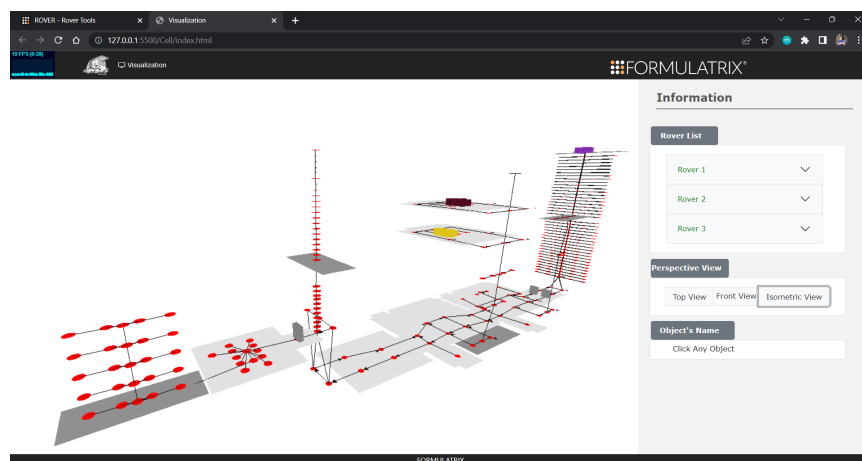
**Gambar 5.27** *Top View*

Pada tombol top view akan menampilkan visualisasi 3D sudut pandang dari atas.



**Gambar 5.28 Front View**

Pada **Gambar 5.28** diatas ditunjukkan gambar dari menu *front view*. Tombol *Front View* akan menampilkan visualisasi 3D dari arah sudut pandang depan.

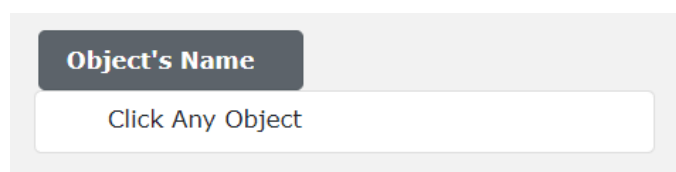


**Gambar 5.29 Isometric View**

Sedangkan pada tombol Isometric View akan menampilkan visualisasi 3D dari sudut pandang samping yang ditunjukkan pada **Gambar 5.29** diatas.

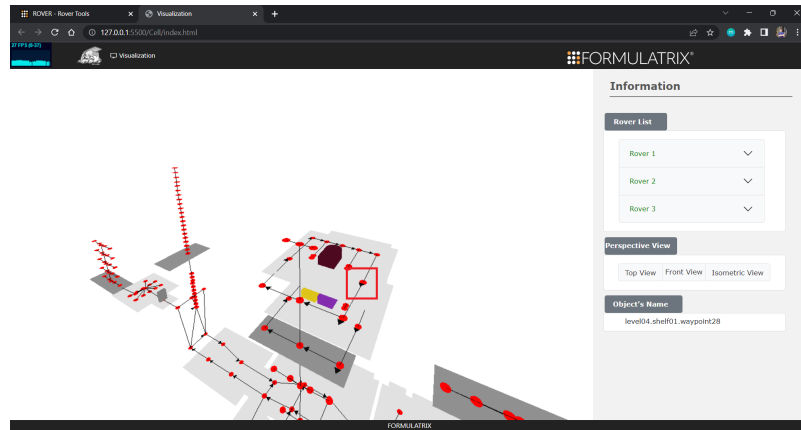
### 5.3.5 Object's Name

Pada **Gambar 5.30** dibawah ini ditunjukkan gambar dari menu Object's Name yang ada pada *side panel*.



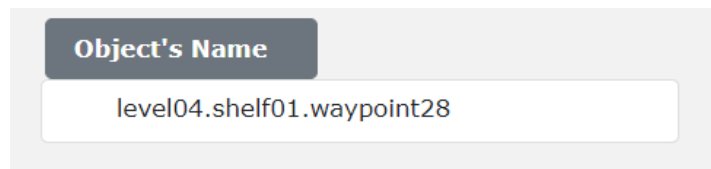
**Gambar 5.30 *Object's Name***

Pada bagian *side panel* terdapat *Object's name* yang dimana ini berfungsi untuk menampilkan informasi dari nama-nama objek yang kita pilih menggunakan kursor. Nama-nama yang dapat ditampilkan yaitu, nama dari robot rover yang aktif, nama *vertices*, nama dari objek seperti *door* dan *elevator*.



**Gambar 5.31 Menampilkan object name pada side-panel**

Seperti pada **Gambar 5.31** diatas, jika kursor diarahkan ke kotak warna merah(vertices), maka pada side panel *Object's Name* akan menampilkan nama sesuai dengan nama vertices yang dipilih.

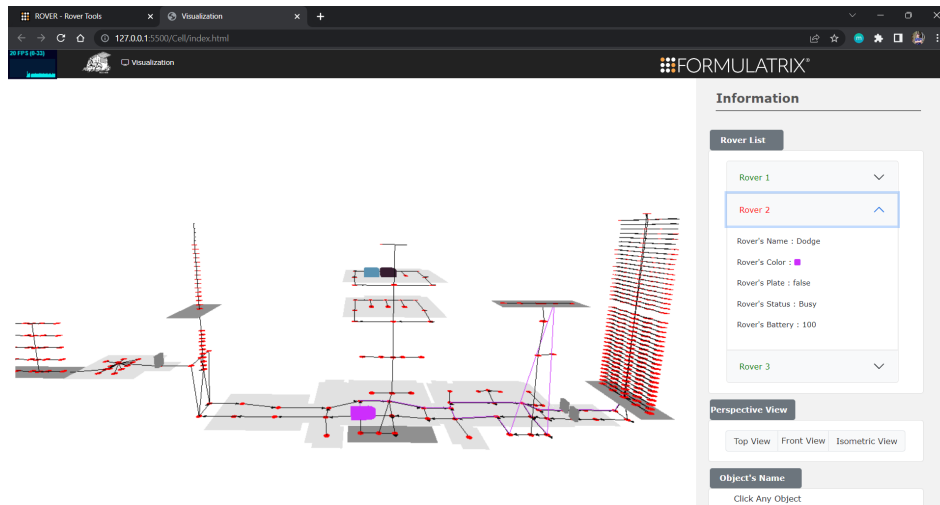


**Gambar 5.32 *Detail object's name***

Salah satu contoh nama dari vertices yang dipilih, yaitu nama yang ditampilkan sesuai dengan nama vertices seperti yang ditunjukkan pada **Gambar 5.32** diatas.

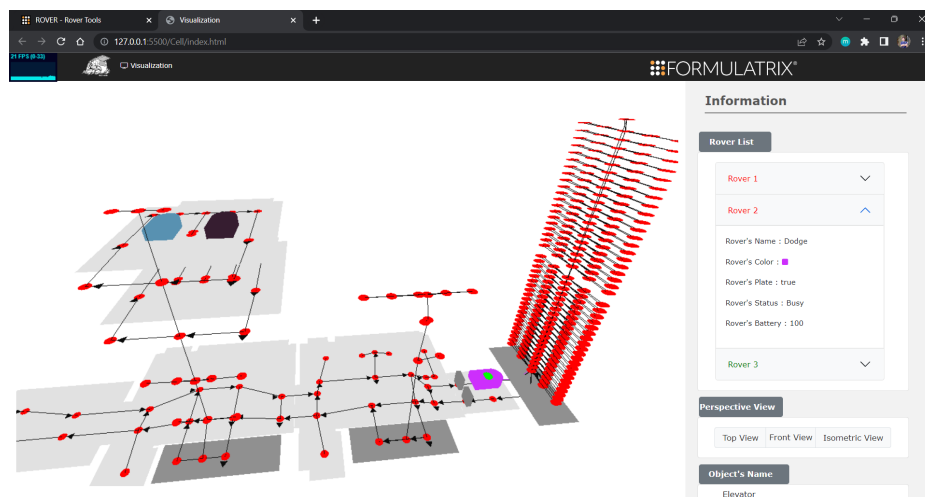
### 5.3.6 Rover Sedang Menjalankan Task

Pada **Gambar 5.33** dibawah ini ditunjukkan gambar pada saat rover sedang menjalankan *task*.



**Gambar 5.33 Rover Berjalan Menuju *Destination***

Gambar diatas menunjukkan rover sedang berjalan menuju *destination*, ditandai dengan adanya garis berwarna ungu sebagai trip yang akan dilalui rover dan pada *side panel* dapat dilihat bahwa warna dari tulisan Rover 2 telah berubah menjadi warna merah yang menunjukkan Rover sedang menjalankan *task*.

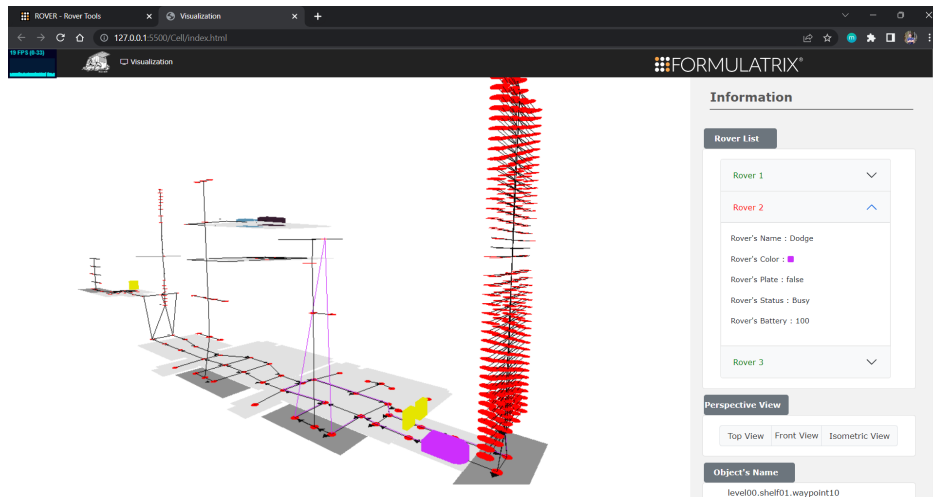


**Gambar 5.34 Rover Sedang Membawa *Plate***

Pada **Gambar 5.34** diatas menampilkan rover sedang membawa *plate* yang ditandai adanya bulatan berwarna hijau diatas rover.

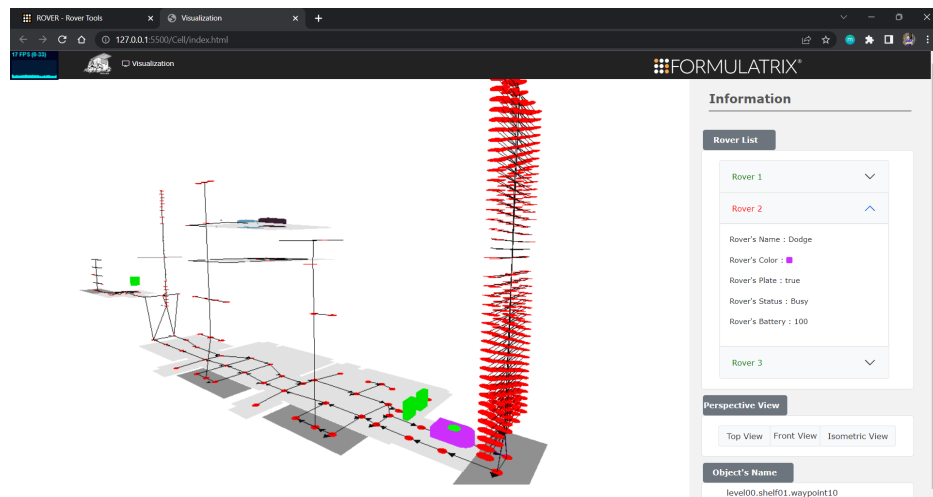
### 5.3.7 Door Open

Pada **Gambar 3.35** dibawah ini ditunjukkan gambar pada saat *door* sedang membuka.



**Gambar 5.35 Door Isopening (sedang membuka)**

Gambar diatas menunjukkan kalau pintu sedang dibuka yang ditandai dengan warna dari pintu yang berubah menjadi warna kuning namun pada proses *isopening* ini, pintu belum terbuka sepenuhnya sehingga rover belum dapat melewati pintu.

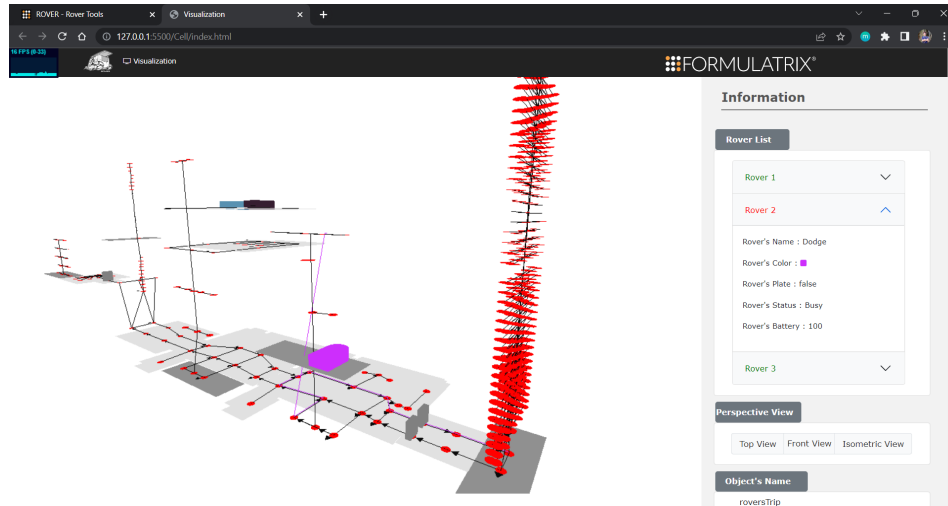


**Gambar 5.36 Door Issettled (terbuka)**

Pada saat *Door Issettled* yaitu keadaan ointu yang sudah terbuka sepenuhnya yang ditandai dengan warna dari pintu yang sudah berubah menjadi warna hijau yang dimana pada keadaan ini rover sudah dapat melewati pintu seperti yang ditunjukkan pada **Gambar 5.36** diatas.



### 5.3.8 Elevator Sedang Bergerak



**Gambar 5.37 Elevator Bergerak**

Pada **Gambar 5.38** diatas menunjukkan elevator yang bergerak dan sedang membawa rover dan pada kondisi ini, rover tidak jatuh melainkan mengikuti elevator turun ataupun naik.

## BAB 6 PENUTUP

### 6.1 Kesimpulan

Setelah proses pengembangan hingga pengujian proyek website simulator pergerakan robot rover selesai, dapat ditarik kesimpulan untuk menjawab rumusan masalah yang sudah ditentukan sebelumnya sebagai dasar pengerjaan proyek ini. Oleh karena itu, berikut adalah hasil kesimpulan yang kami peroleh:

1. Penulis berhasil mengembangkan website simulator robot Rover dengan tampilan 3D sebagai bentuk pengembangan dari tampilan sebelumnya yang dikembangkan oleh tim developer instansi dimana tampilan masih dalam bentuk 2D.
2. Website simulator robot Rover diprogram secara generated dengan....., dan bukan menggunakan manual-code. Sehingga visualisasi robot Rover beserta komponen-komponennya dapat terbentuk secara otomatis menyesuaikan topologi jenis apapun.
3. Pada visualisasi hasil pengembangan, Rover mampu menjalankan task/instruksi yang diberikan oleh user tanpa ada bug maupun error?

### 6.2 Saran

Berdasarkan seluruh pengujian pada proyek ini dilakukan, terdapat beberapa kekurangan dari proyek ini. Maka dari itu, berikut beberapa saran untuk menyempurnakan dan juga memperbaiki kekurangan dari proyek ini:

1. Pada penelitian ini masih menggunakan jaringan localhost. Pada pengembangan penelitian selanjutnya dapat membangun sistem dengan menggunakan jaringan internet dengan jaringan LAN.
2. Animasi pada website visualisasi masih kurang mulus karena pergerakan rover dalam visualisasi hanya di update setiap satu detik.
3. Untuk pengembangan visualisasi selanjutnya agar lebih baik menggunakan library tambahan selain *library ThreeJs* seperti *Babylon.js* untuk menghasilkan animasi yang lebih baik lagi.
4. Pada penelitian ini, hanya terdapat fitur detail informasi terkait rover yang sedang berjalan secara lengkap. Mungkin pada pengembangan visualisasi selanjutnya agar dapat digabung dengan website fleet manager yang berfungsi untuk memberikan *task* kepada rover.

## DAFTAR PUSTAKA

- Bootstrap, 2011. *Bootstrap · the most popular html css and js library in the world*. [online] Tersedia di : <https://getbootstrap.com/>.
- Formulatrix, 2022. *About FORMULATRIX Laboratory Automation Solutions*. [online] Tersedia di : <https://formulatrix.com/about-us/>.
- Pohan, M. R., Trilaksono, B. R., Santosa, S. P., & Rohman, A. S. (2019). Algoritma Perencanaan Jalur Kendaraan Otonom di Lingkungan Perkotaan dari Sudut Pandang Filosofi Kuhn dan Filosofi Popper. *TELEKONTRAN, VOL. 7, NO. 2*.
- ThreeJs, 2010. *Three.js - Javascript Library*. [online] Tersedia di: <https://threejs.org/>.
- Wiradarma, G. R., Piarsa, I. N., & Putra, I. G, 2017. Media Pengenalan Properti 3D Berbasis Web Aplikasi. *MERPATI VOL. 5, NO. 1*.
- Nandi, Amit Kumar and Pattanaik, Laxmi Narayan, 2020. *Design and Development of a Web-Based Robotics Simulator*. Proceedings of the International Conference on Recent Advances in Computational Techniques (IC-RACT) 2020, Tersedia di SSRN: <https://ssrn.com/abstract=3697570>
- Huang, D., XingXu, 2022. *IGAOD: An online design framework for interactive genetic algorithms*. Science Direct: Vol.19, pp. 25-26, [online] Tersedia di : <https://doi.org/10.1016/j.softx.2022.101205>.
- A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J.P. Laulajainen, et al., 2022. *Platform for distributed 3D gaming*. Int J Comput Games Technol, pp. 1-15.

## LAMPIRAN DOKUMENTASI KEGIATAN PROGRAM PKL



