Robotics Middleware Framework

RoMi-H Development to Deployment

WEBINAR IS STARTING SOON..

Centre for Healthcare Assistive & Robotics Technology

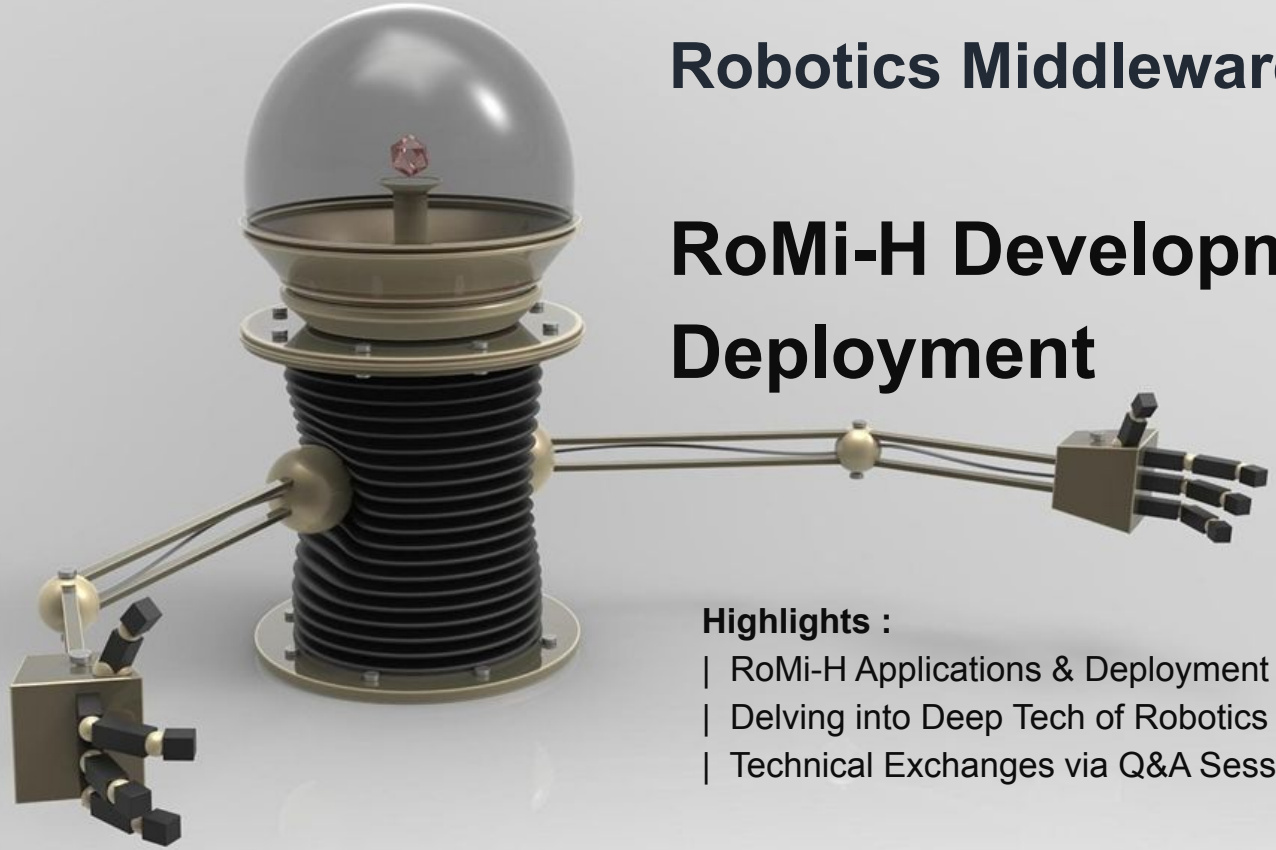
IHiS Because IT's in Healthcare

HOPE technik

open robotics

Catalysed by MINISTRY OF HEALTH SINGAPORE

NATIONAL ROBOTICS PROGRAMME



Robotics Middleware Framework

RoMi-H Development to Deployment

Highlights :

| RoMi-H Applications & Deployment in Real-live Environments
| Delving into Deep Tech of Robotics Middleware
| Technical Exchanges via Q&A Sessions

Centre for Healthcare Assistive & Robotics Technology

IHiS Because IT's in Healthcare

HOPE technik

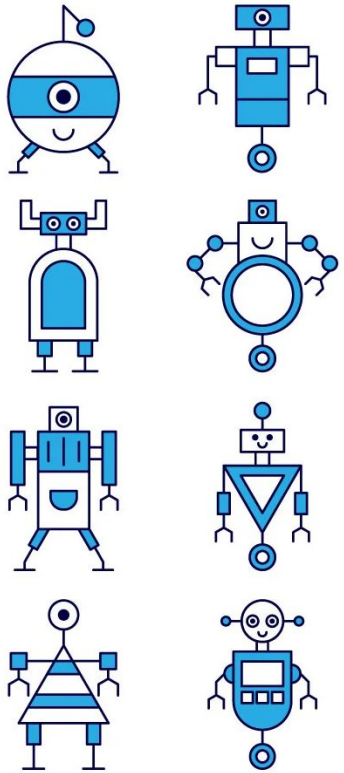
open robotics

Catalysed by MINISTRY OF HEALTH SINGAPORE

NATIONAL ROBOTICS PROGRAMME

Webinar Itinerary (1/2)

TOPICS	SPEAKERS
Opening Address	Selina Seah ACEO of Changi General Hospital & Director of CHART
Care Treatment Facilities - **Changi General Hospital Deployment**	Alphonsus Tay Robotics Engineer, CHART
Care Treatment Facilities - **EXPO Halls Deployment**	Christopher Lau Senior Software Engineer, Hope Technik Pte Ltd
Deployment beyond Healthcare Facilities	
Q&A Session One	
Break Time	

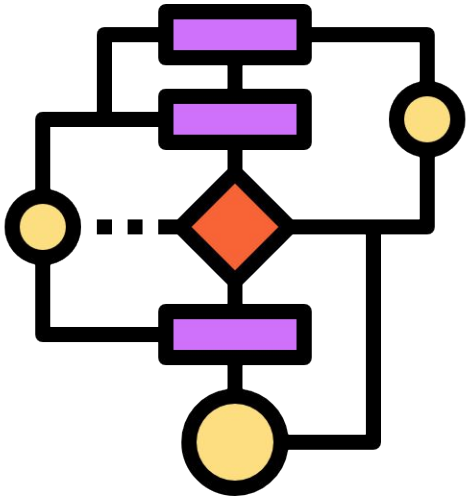


Centre for Healthcare Assistive & Robotics Technology

Catalysed by MINISTRY OF HEALTH SINGAPORE

Webinar Itinerary (2/2)

TOPICS	SPEAKERS
Traffic Editor III	Dr Morgan Quigley Chief Architect, Open Robotics Singapore
Indoor/Outdoor Operations	
FreeFleet	
Adapters and Templates	Dr Michael Grey Software Engineering Supervisor, Open Robotics Singapore
Flexible Task System + Teleoperation	
RMF Web + Release Review	
Q&A Session Two	
Closing Address	Prof Quek Tong Boon Chief Executive, National Robotic Programme



Journey of Robotics Middleware Framework

Beginning

Concept of RMF was born

Party Up

Consortium was formed with clear visions

Demo Platform 1

First build of RMF working with simple Robots Controls, Sensors and Actuators

Demo Platform 2

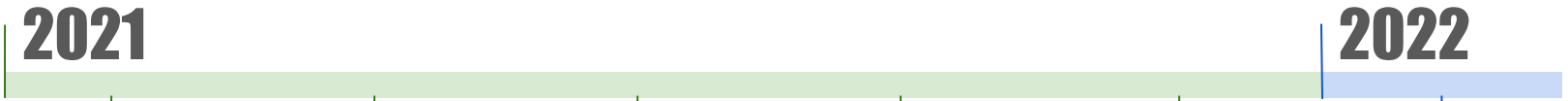
Integration of Medical Devices, Building Infra and Deconfliction of Robots via RMF

1st Webinar!

Livestream of RMF enabling interoperability between robots, infra and devices

SRB Clearance

Approval in IHIS Solution Review Board to implement RMF in SG Public Hospitals



Teleops through 5G Technologies

RMF remote robot operations capabilities through secured VPN connection over 5G

RMF-on-Cloud and Cybersecurity

Leverage on cloud technologies for rapid deployment and stepping up on cybersec measures

Deputy PM Showcase

Demonstrated RMF-Anywhere capabilities through 4G teleops @NTU

Demo Platform 3 @ Sentosa Island

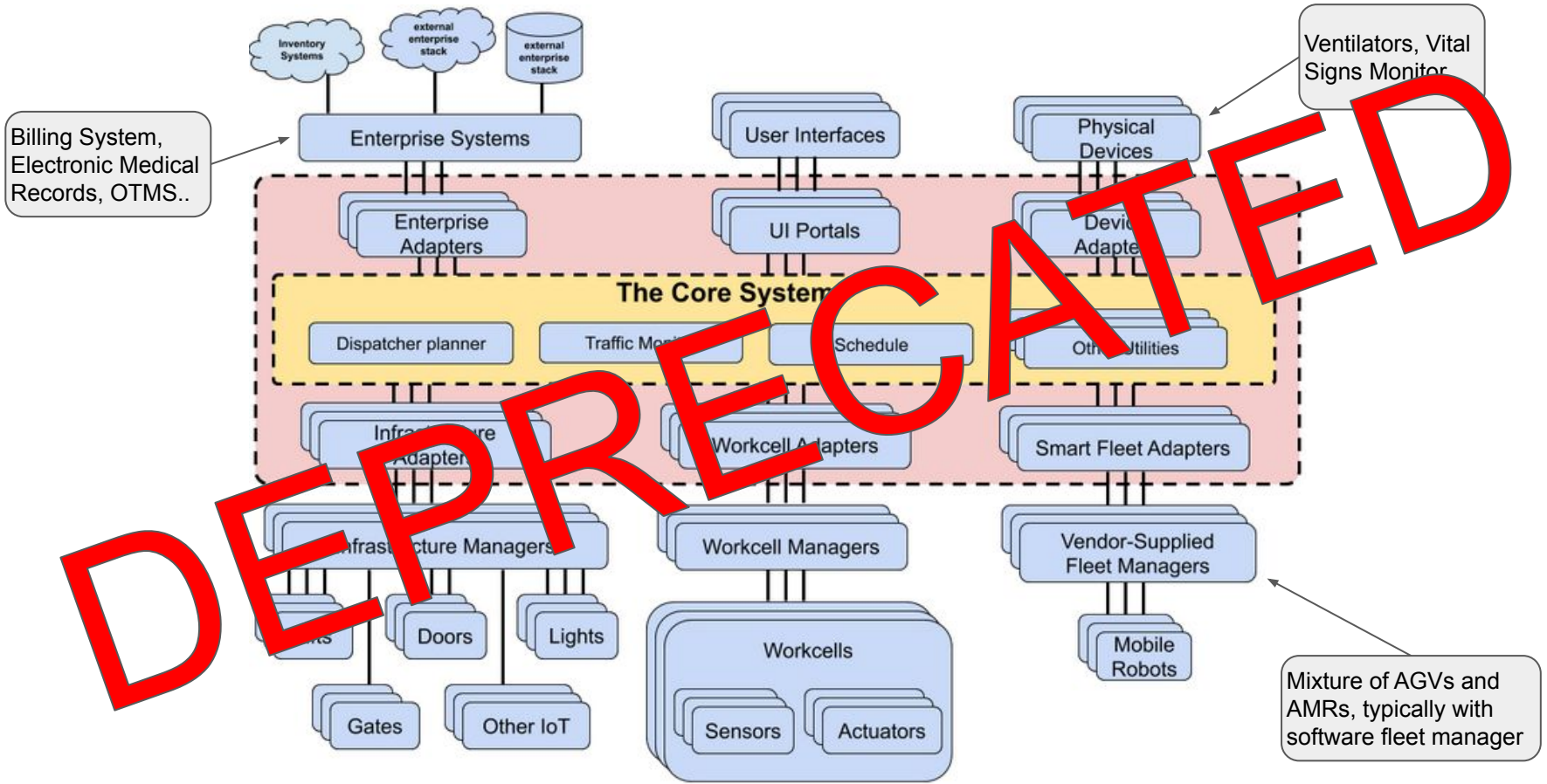
Mass Casualty Scenario involving organisations such as SingTel, GovTech, MOH, CGH, with Field Medical Teams and real(!) ambulances

Singapore Standards TR93:2021 Published

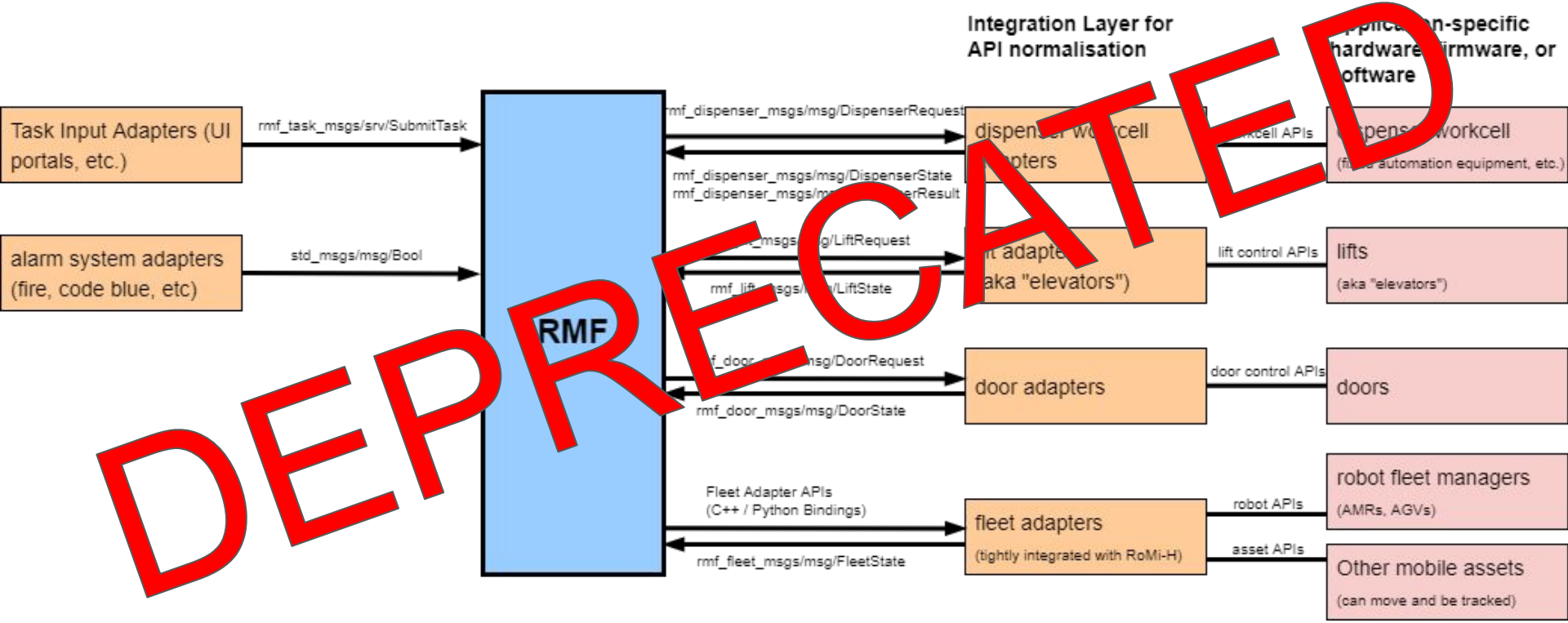
Established industrial standard with members such as ESG, A*Star, IMDA, Kone, Beckoff, TÜV SÜD, SIIX-AGT, Sesto Robotics, NTU, ROS-Industrial etc

Adoption of Enterprise Systems in Healthcare and other industries as part of Smart Buildings/Operations

Initial RMF Architecture Diagram



Standardised Interfaces and Messages (*end 2019*)



DEPRECATED

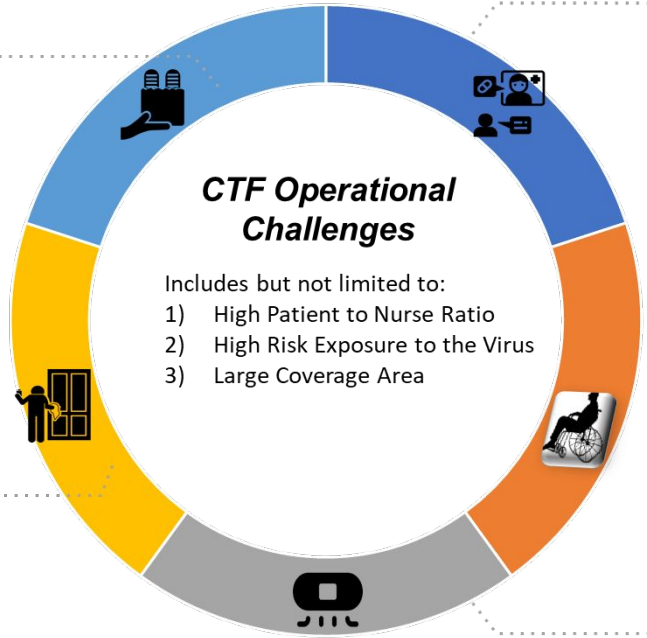
Approach to Operational Challenges of Site Deployment

Scheduled/Ad-hoc Deliveries to Patient Bedside

Meals and supplies can be delivered to patients bedside. Eliminates need for Care Team to gather supplies from Ward Store and distributing it to patients.

Disinfection

Manual disinfection may be inadequate and subject to human error, especially in such a large space; Moreover, it tends to be cumbersome and labour intensive



Patient-Care Team Interaction

Reduce number of care personnel needed to do surveillance. Care Team can interact and intervene when abnormalities are detected .

Aged Care

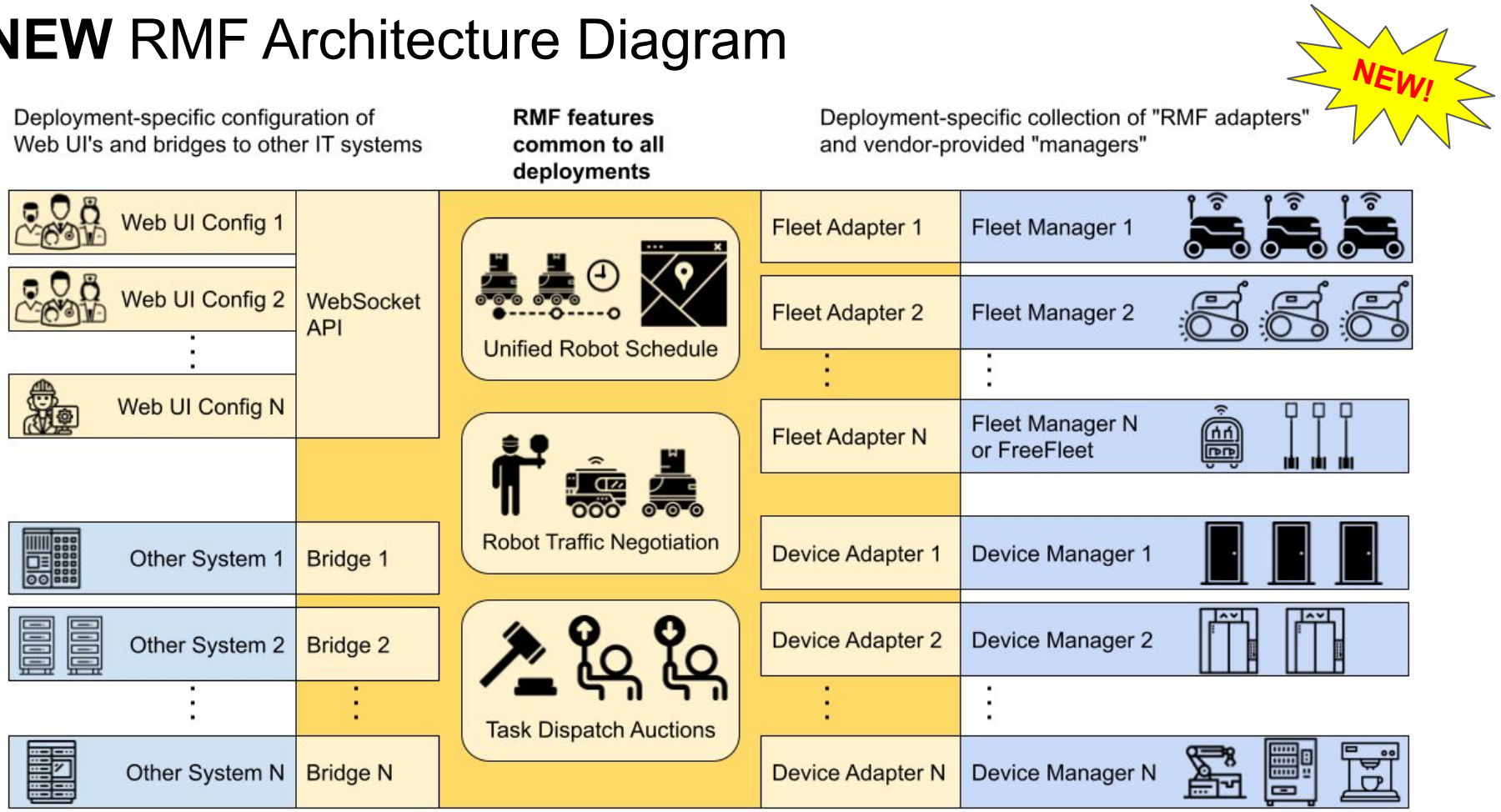
Residents may have mobility challenges or limitations which require assistance from nurse/HCA to the washrooms or common facilities. Care personnel may have to make multiple trips to the rooms, taking time away from their core activities

Cleaning

Manual cleaning will be highly labour intensive and may leave some neglected areas such as floor corners and edges;

Centre for Healthcare Assistive & Robotics Technology
IHiS
HOPE technik
open robotics
Catalysed by MINISTRY OF HEALTH SINGAPORE
NATIONAL ROBOTICS PROGRAMME

NEW RMF Architecture Diagram

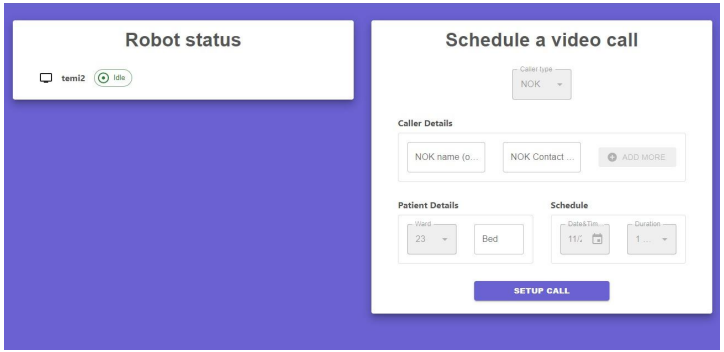


CTF - Medical Intensive Care Unit Deployment (1/2)

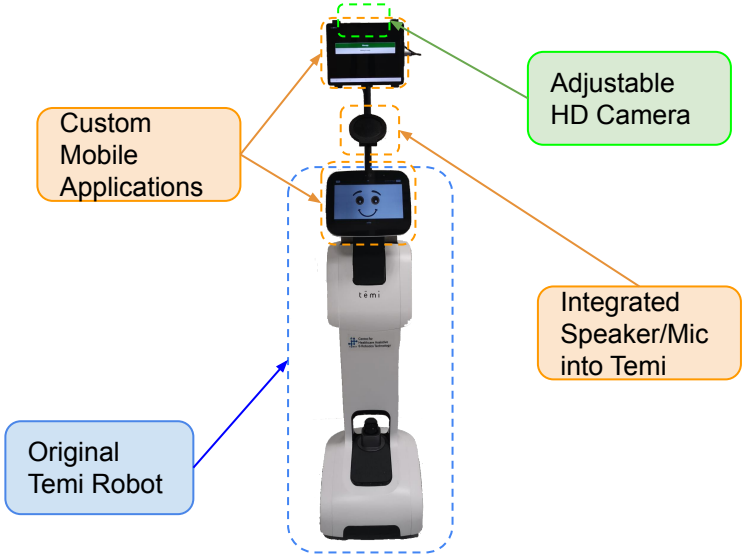
Problem Statement:

MICU Care Team constantly help patients in setting up video calls with patient's loved ones and stay "rooted" to holding mobile devices for long periods of time

Solution:



(1) In-house developed UI on Temi and Mobiles, together with backend teleconferencing services



(2) Lots of Engineering!

CTF - Medical Intensive Care Unit Deployment (2/2)

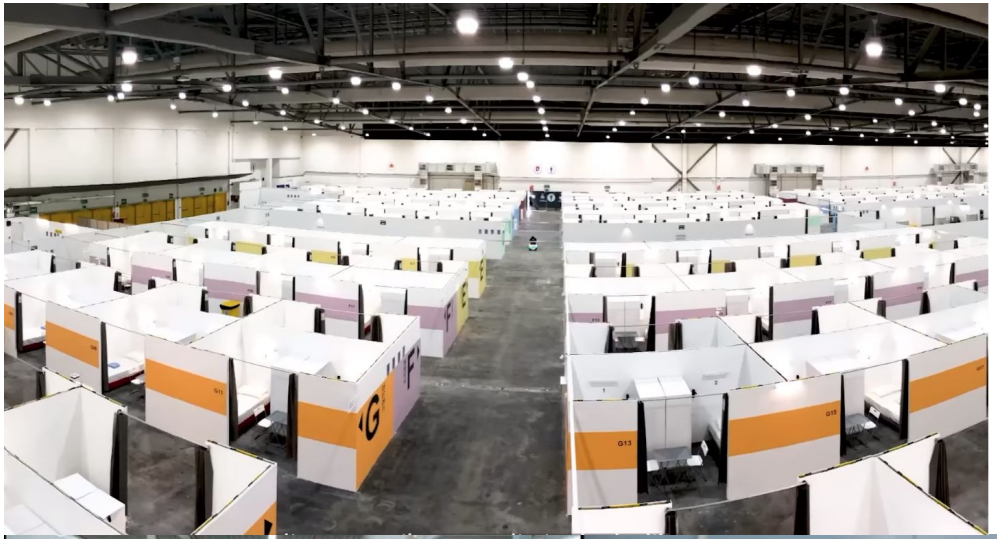
Solution Functionalities:

- Automatic and seamless connection between Telepresence Robot and patient's Next-of-Kin
- Scheduled autonomous navigation of Telepresence Robot to patient bedside
- Remote Control of Telepresence Robot by Next-of-Kin *(e.g change camera angles/views)*

Benefits:

- Free up Care Team for other patient needs
- Patients enjoys more privacy
- Reduce unnecessary staff exposure

CTF - Expo Deployment

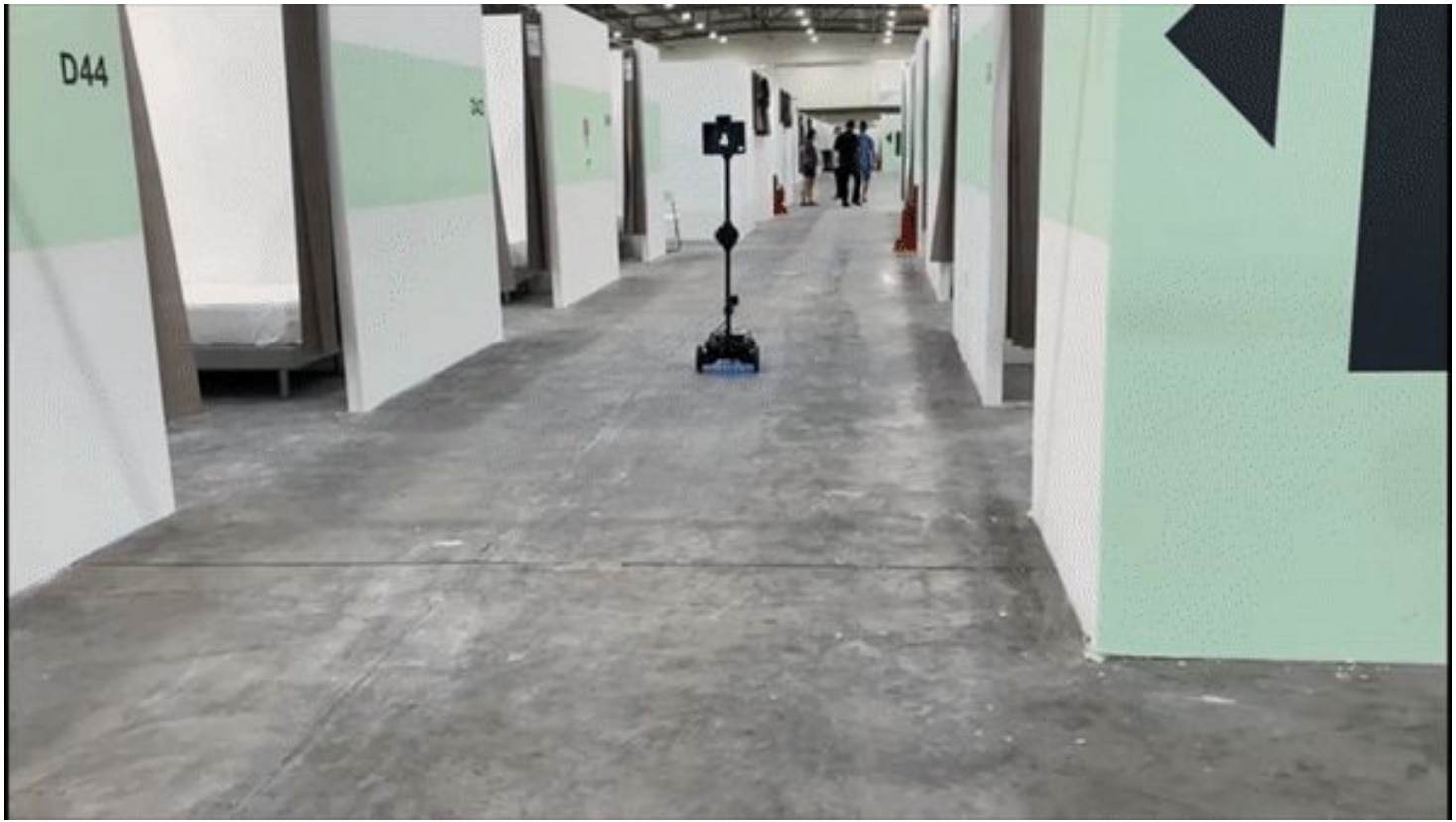


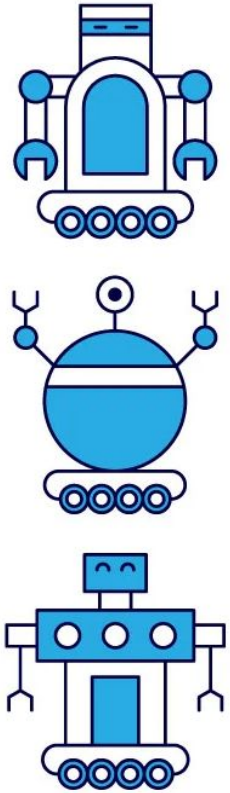
- Rapid Deployment in Hall 9 & 10 in less than one week
- Enabling care team and logistics workflows for 700+ beds
- Delivery bot, Disinfection bot, Telepresence bot **interoperable and <u>unified</u>** under RoMi-H

Video source:
Mr Ong Ye Kung *(Minister for Health of Singapore)*

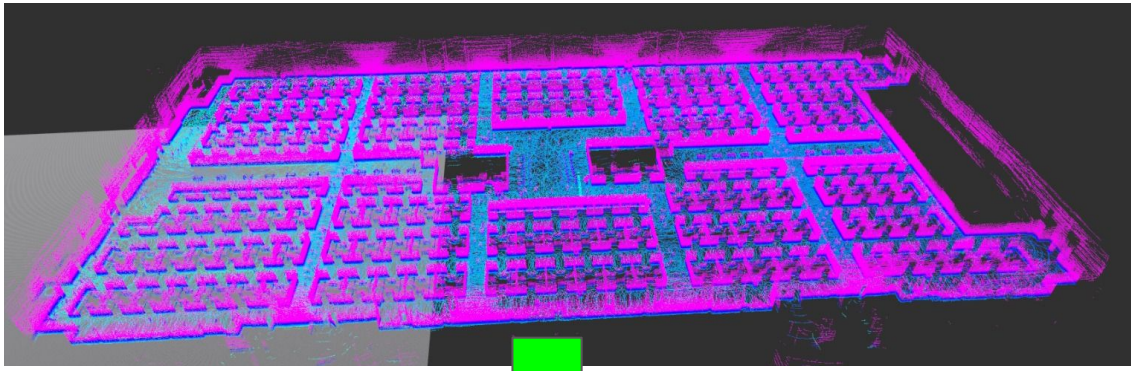
Top: Bird's eye view of Expo Hall 9 | **Bottom Left**: Magnus collecting bags of dirty linen | **Bottom Middle**: Healthguard misting liquid ozone solution | **Bottom right**: Temi navigating autonomously to patient's bedside for a video call

Different robots working together in same space



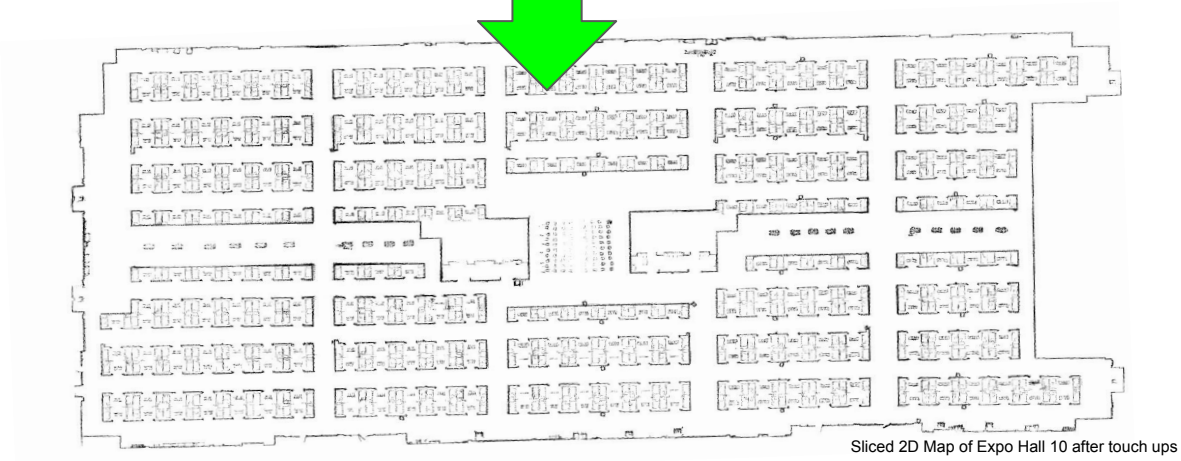


Robotic Agnostic Mapping Platform (RAMP)



3D Map of Expo Hall 10

3D sliced to 2D



Sliced 2D Map of Expo Hall 10 after touch ups

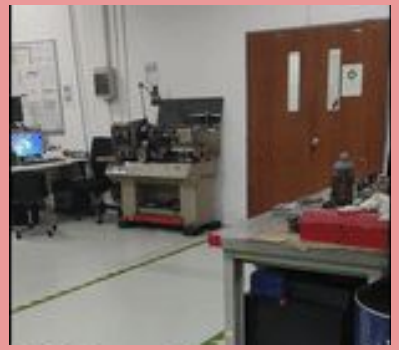
- 3D map sliced into multiple layers of different height
- Corresponding 2D map fed into its navigation stack based on lidar height on robot
- Reduce deployment time
- Reusable by other robots

RAMP-Bot mounted with velodyne LIDAR and display monitor to visualize SLAM

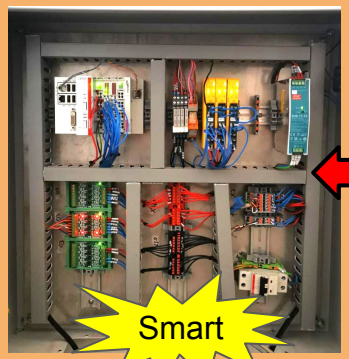
Deployment beyond Healthcare Facilities

- “Smartify” existing doors & cargo lift
- Building our own Test farm right in our complex, enabling faster integrations
- 25 year old lift - works with RMF!
- RMF going to Libraries, Airport, Seaport, Commercial Buildings and more..

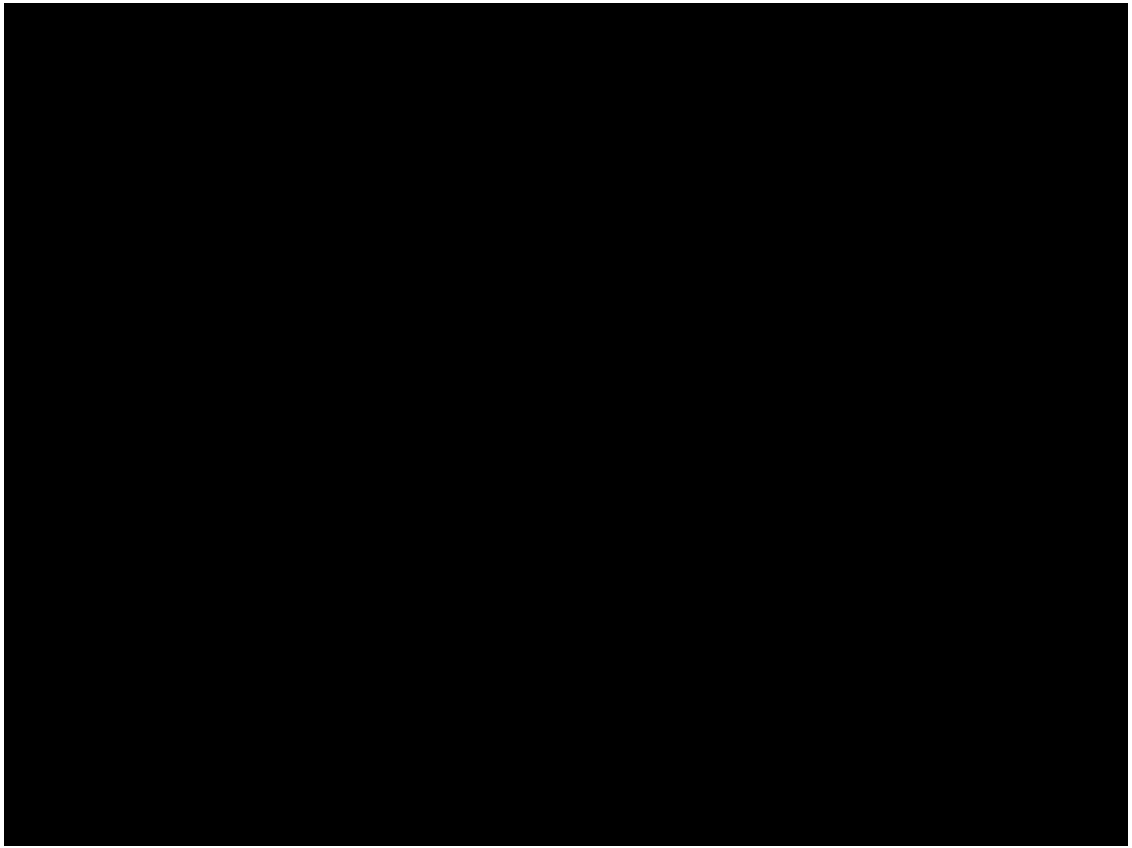
RMF Enhanced Door



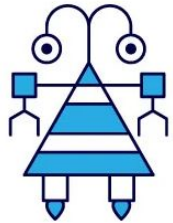
RMF Enhanced Lift

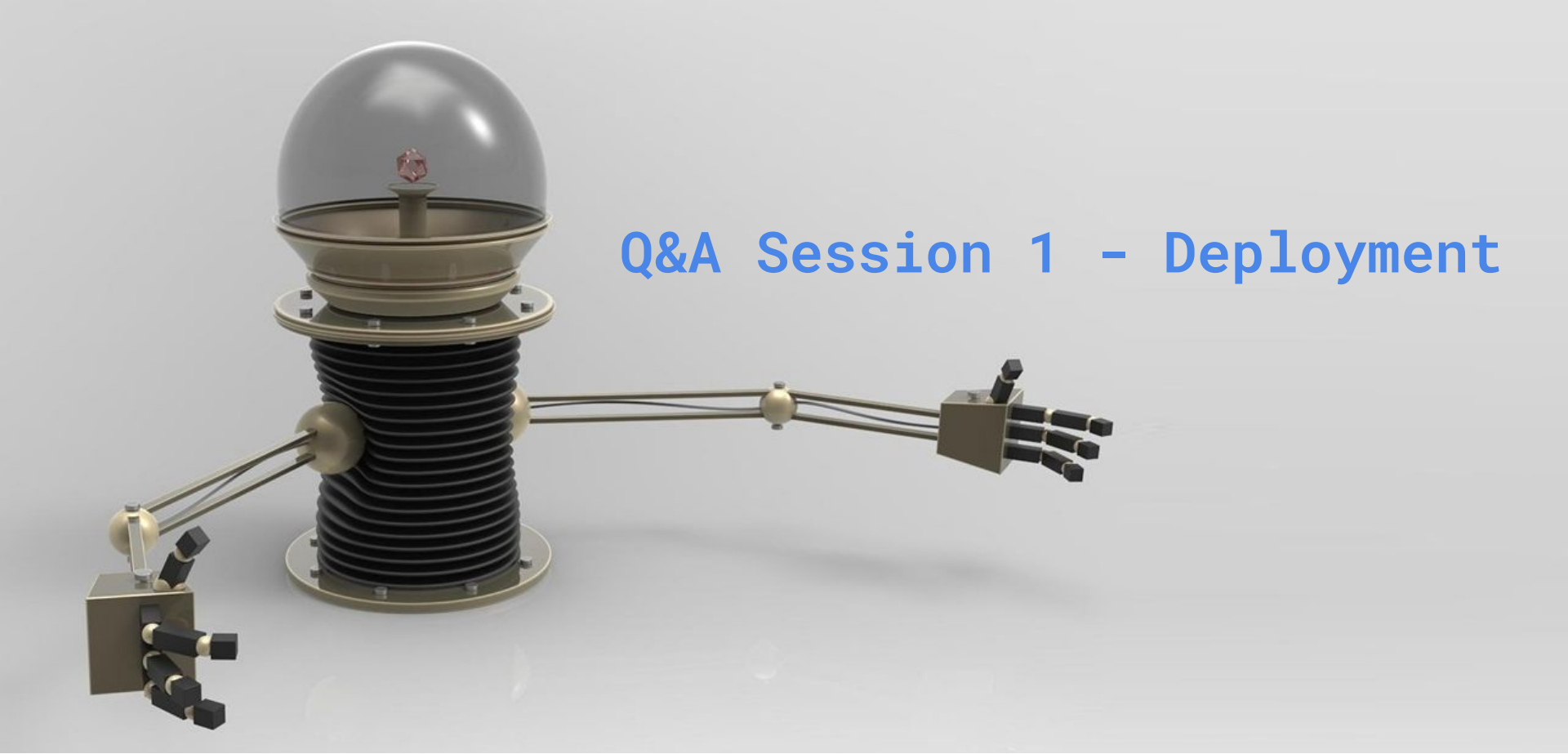


RMF and Robots: How does it work?



- Capable of Multi-Floor, Multimap operations
- Many robot info available, including battery status
- Door states, Lift states etc
- Task ID, last updated etc
- Projected paths and parking lots for de-conflictions





Q&A Session 1 - Deployment

Centre for Healthcare Assistive & Robotics Technology

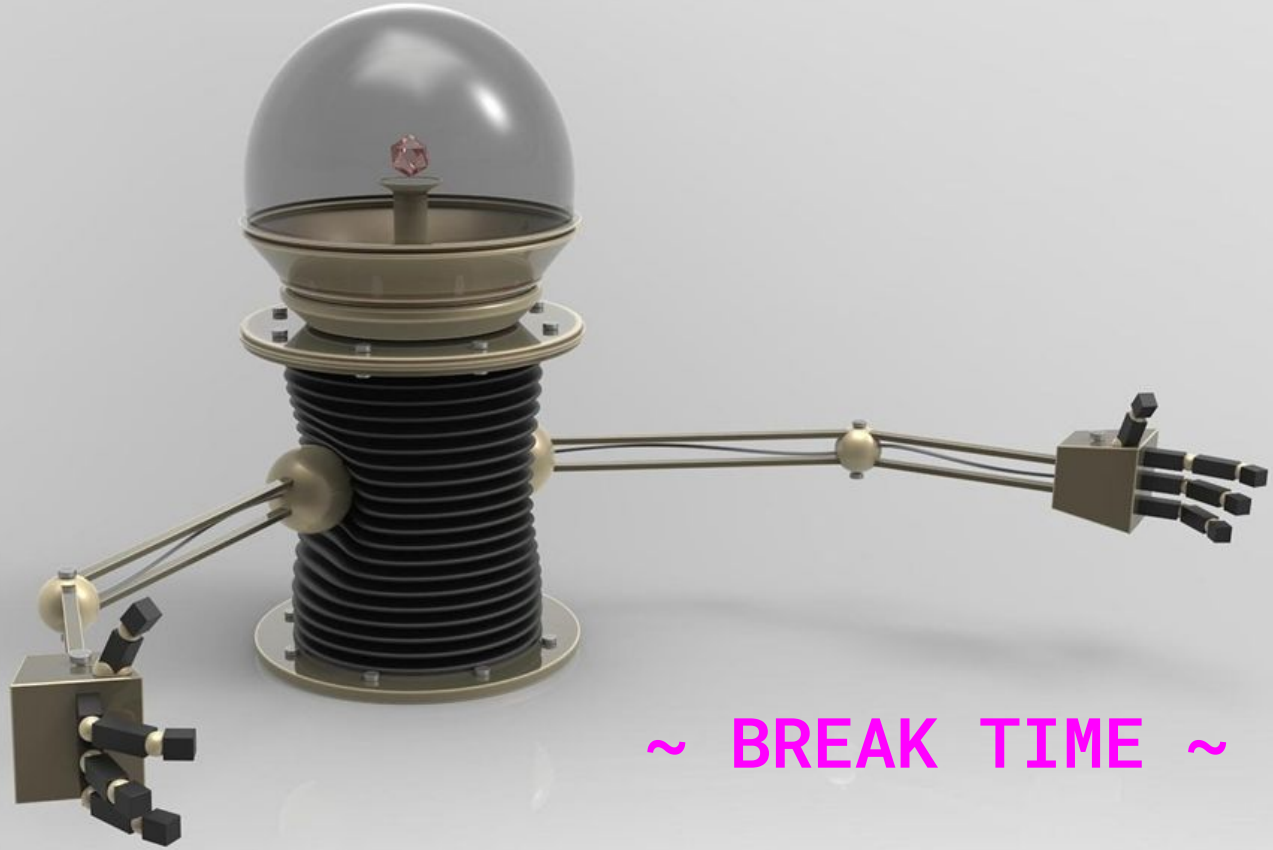
IHiS Because IT's in Healthcare

HOPE technik

open robotics

Catalysed by MINISTRY OF HEALTH SINGAPORE

NATIONAL ROBOTICS PROGRAMME

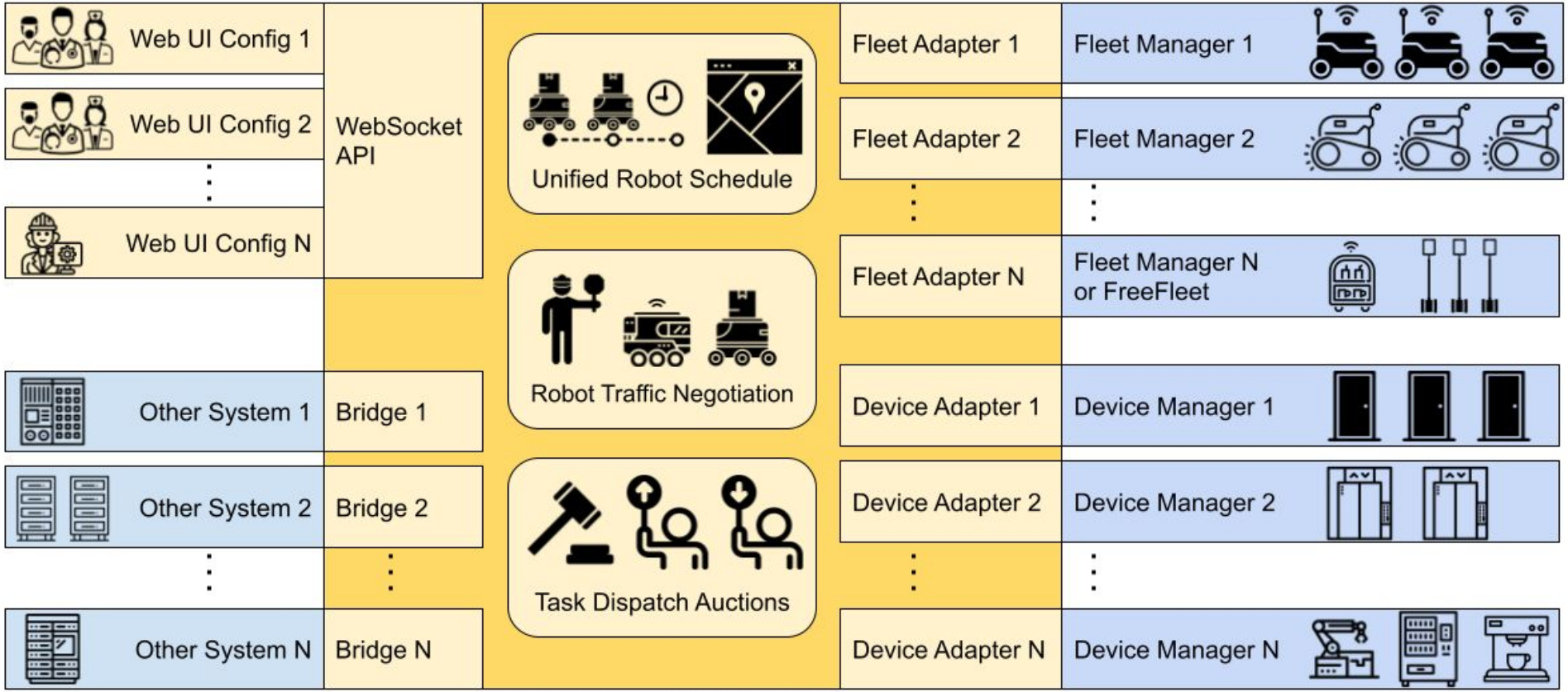


~ BREAK TIME ~

Deployment-specific configuration of Web UI's and bridges to other IT systems

RMF features common to all deployments

Deployment-specific collection of "RMF adapters" and vendor-provided "managers"

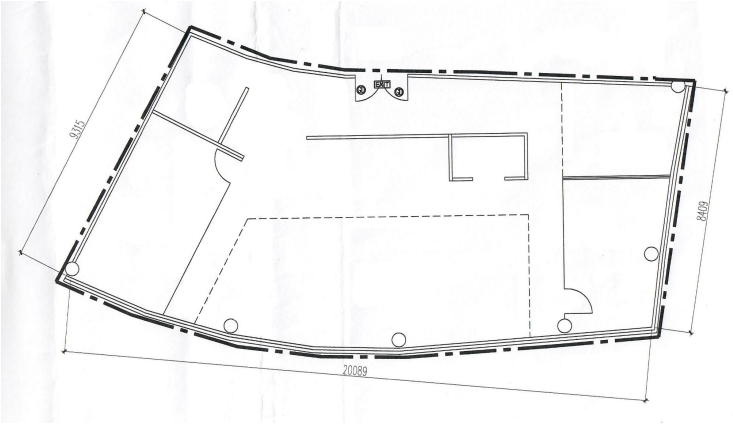


Traffic Editor

- Semi-automatic map alignment
- Sneak peek at Traffic Editor III

https://github.com/open-rmf/rmf_traffic_editor

Aligning robot-generated maps to floorplans



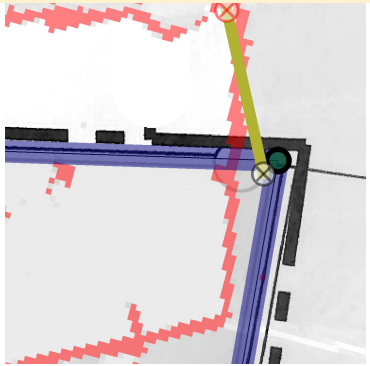
Building floor plan: the reference coordinate system

Robot-generated map

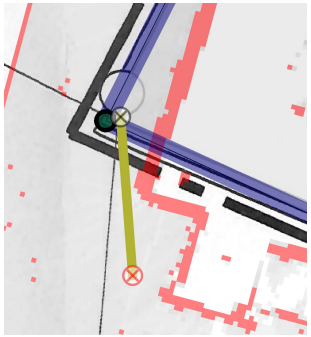
- Until recently, it was necessary to manually edit transformation parameters in Traffic Editor until it "looked OK"
- It worked
- It was painful
- ***Now, we shall introduce Semi-Automatic Alignment!***

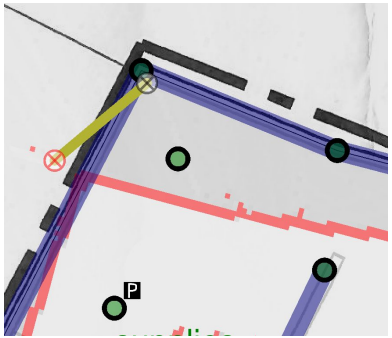
https://github.com/open-rmf/rmf_traffic_editor

Semi-automatic map alignment

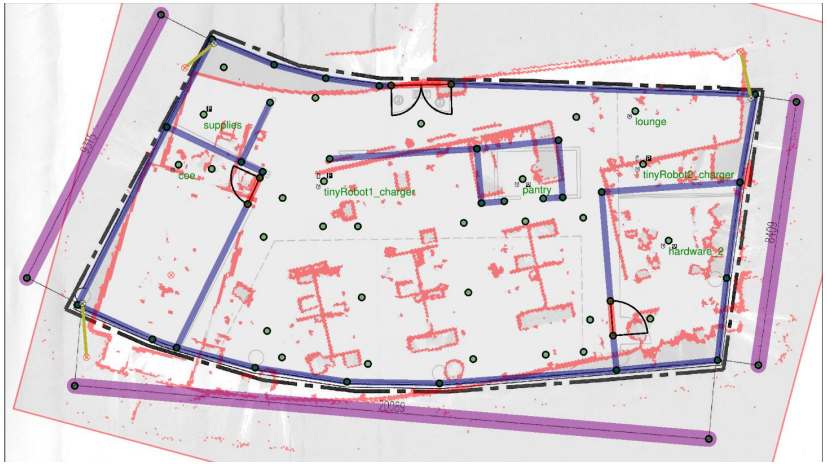


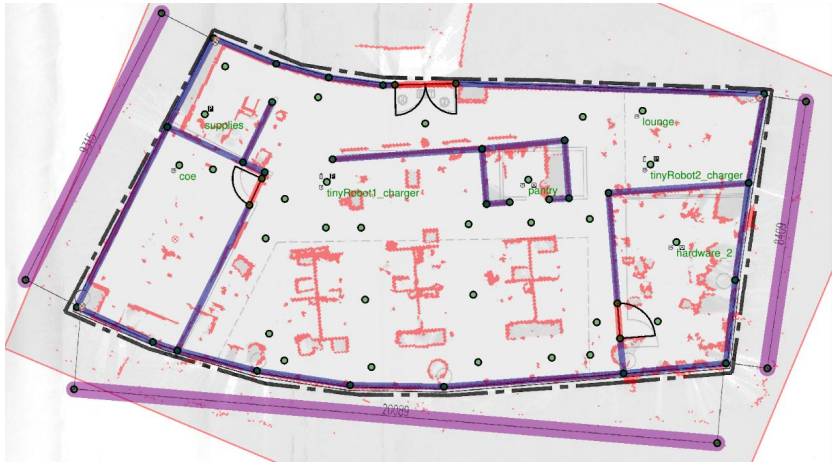
- Fully-automatic alignment is hard
  - As-built often differs from floorplan, especially non-load-bearing walls
  - Cabinets, pillars, chairs, etc., can occlude walls
- Humans can easily spot these issues and ignore them
  - Computers cannot
- "Semi-Automatic" alignment workflow:
  - Roughly align the robot map using manual transform parameters
  - Click "features" that you can see in both robot map and floorplan
    - corners of load-bearing walls
    - pillars
  - Click "constraints" between corresponding features
  - Select Edit->"Optimize Layer Transforms" (Ctrl+T)
    - Numeric solver tries to find a transformation that minimizes the constraint lengths



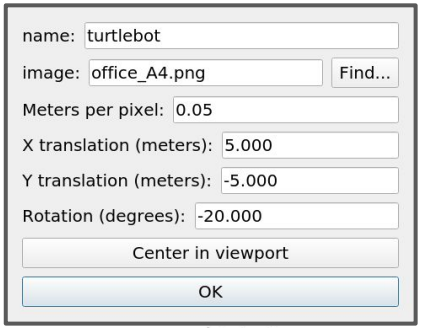


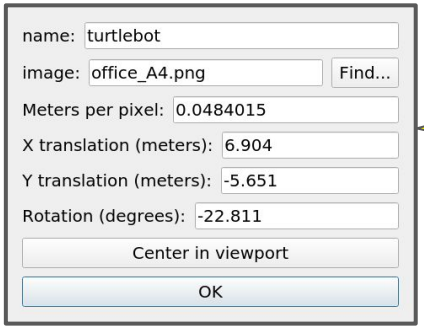
https://github.com/open-rmf/rmf_traffic_editor





Semi-automatic map alignment

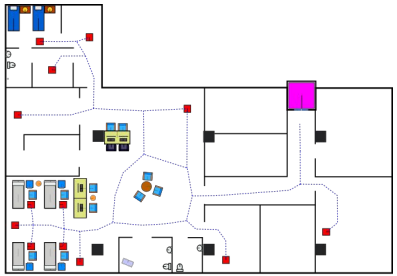




LESS PAIN!

https://github.com/open-rmf/rmf_traffic_editor

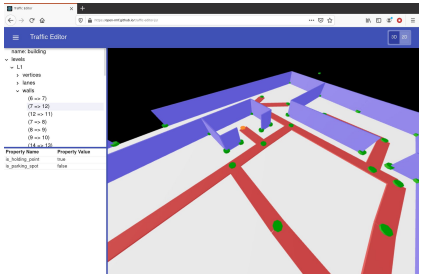
Traffic Editor: Evolution



2018
Inkscape (SVG)
with annotations

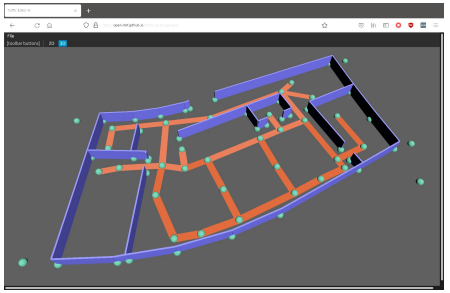
2019
traffic-editor I
C++ / Qt

desktop only



2021
traffic-editor II
TypeScript / THREE.js

web only



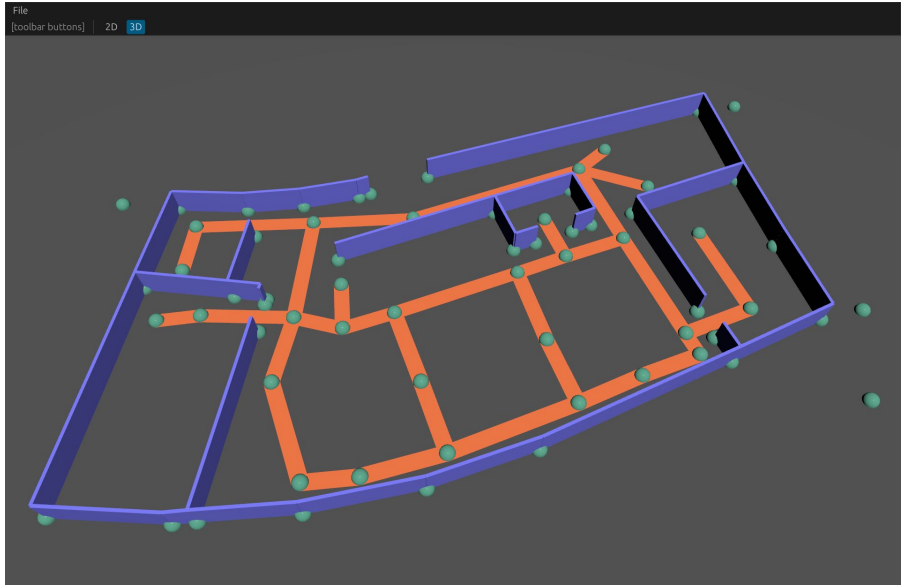
2022
traffic-editor III
Rust / Bevy

desktop and web

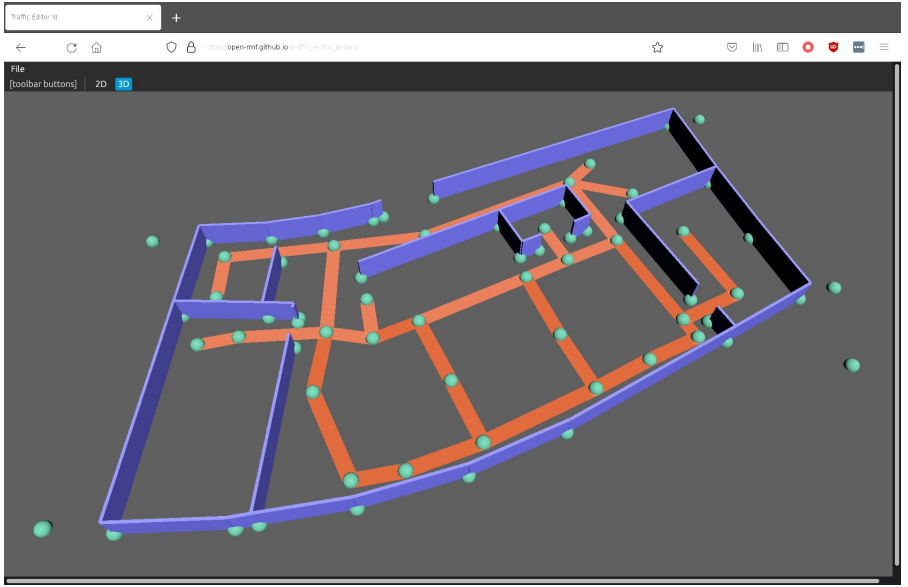
https://github.com/open-rmf/traffic_editor_iii

Traffic Editor III:
Grand Unified Web or Native Application = **GrUWorNaA**

FANCY!



Running on desktop with full GPU acceleration, convenient for R&D and initial deployment setup



Running in web browser for maximum portability, convenient for deployment maintenance and ops

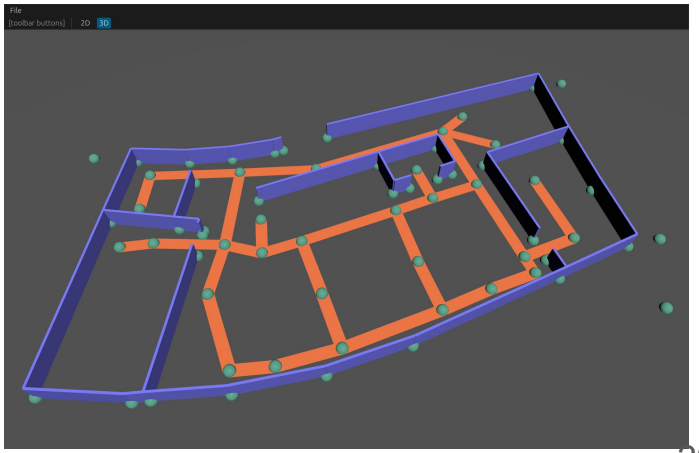
Centre for Healthcare Assistive & Robotics Technology
IHiS
HOPE technik
open robotics
Catalysed by MINISTRY OF HEALTH SINGAPORE

https://github.com/open-rmf/traffic_editor_iii

Traffic Editor III: Some Details

- Written in Rust using the Bevy game engine. Extremely fast (easily 1000's of vertices/edges).
- **native:** compiles to x86_64 code for maximum GPU awesomeness on Vulkan, multithreading, etc
- **web:** compiles to WebAssembly, runs in any browser on WebGL2, but with the usual browser-sandbox performance tradeoffs
- As in Traffic Editor II, has both 2d and 3d modes
- UI widgets via eGUI (successor of Dear ImGui)

Try the webGL version in your web browser!
https://open-rmf.github.io/traffic_editor_iii/



Traffic Editor III: Roadmap

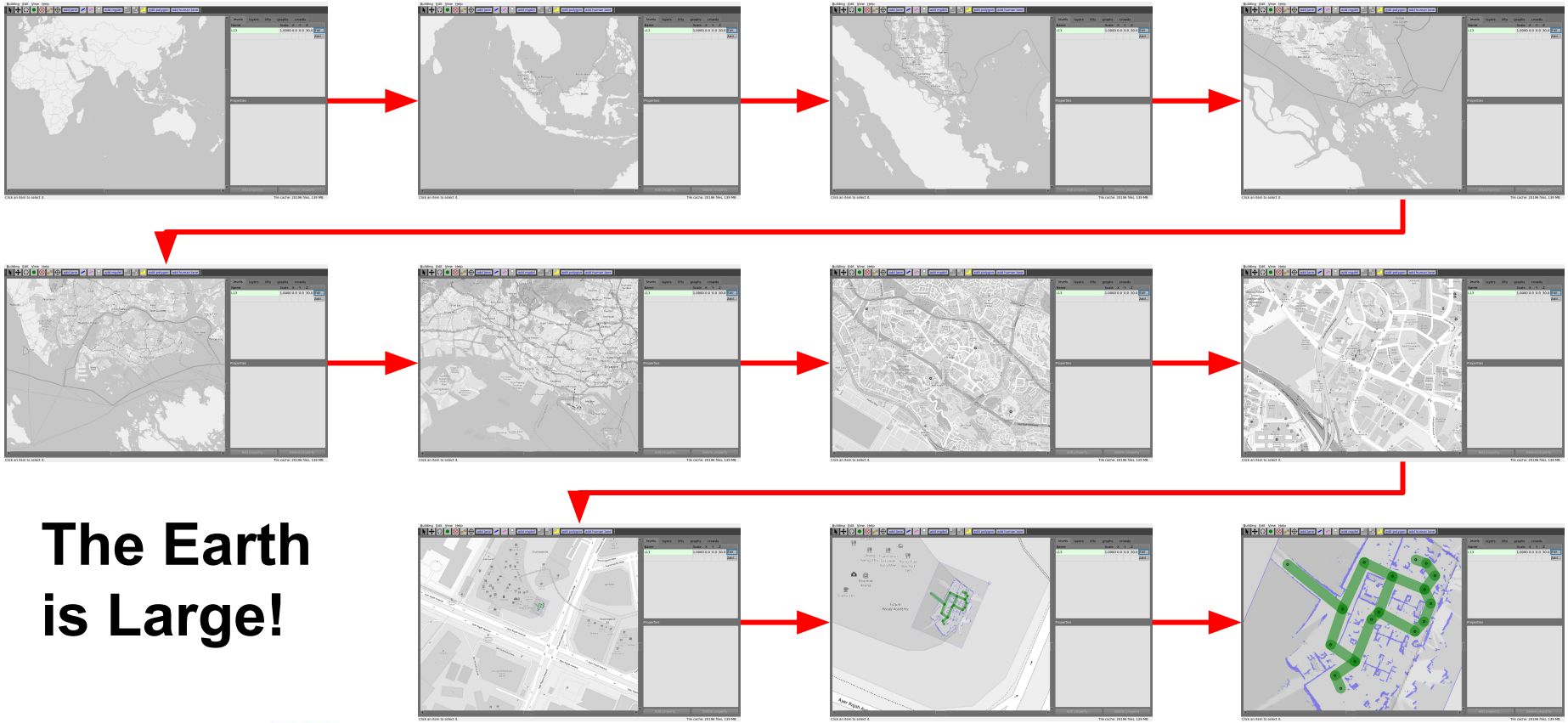
- Proof of concept demonstrates the tech stack seems viable 🎉
- First goal: start using it day-to-day on desktop 🖥️
- Near-term feature development: 👷
  - load and save files from local disk 💾
  - UI sidebar to show/edit parameters of vertices/edges/etc 🪟 ✍️
  - use mouse to add/delete/edit vertices and edges 🖱️
- Will re-evaluate feature needs after those are implemented 🚧

https://github.com/open-rmf/rmf_traffic_editor

Indoor/Outdoor Operations

https://github.com/open-rmf/rmf_traffic_editor

Indoor/Outdoor Operations: Defining traffic in global coordinates



The Earth is Large!

https://github.com/open-rmf/rmf_traffic_editor

Indoor/Outdoor ops: extending the map file format

```
coordinate_system: wgs84
...
levels:
  L13:
    vertices:
      - [103.7879076584487, 1.298879948498967, 0, ""]
      - [103.787955969507, 1.298840697150378, 0, ""]
      - [103.7879749041004, 1.298881757958285, 0, ""]
      - [103.7[illegible]97557543, 1.298869439715983, 0, ""]
```

NEW!

longitude!

latitude!

Indoor/Outdoor ops: Return to Flatland

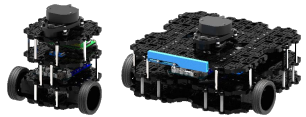
- We don't need or want to run RMF in ellipsoidal (latitude/longitude) coordinates
  - The math is gross and CPU-intensive
  - The world is (basically) flat, at least for robots that don't go far
  - We want a convenient Cartesian approximation for RMF
- Map projection is a solved problem :)
  - Each region has a standard scheme with an EPSG code
  - For example, Singapore's projection is defined in EPSG:3414
- Traffic Editor lets you define the preferred projection scheme for your latitude/longitude traffic map
  - The navigation graphs and simulation models generated from the traffic map will be projected and translated as requested
  - This is "lossless" since global lat/lon coordinates can always be re-calculated as needed
  - Mathematics is done by PROJ, the canonical software for geodetic transforms (lol don't try to write this yourself)
- **Summary: maps are defined in latitude and longitude, then RMF operates in a "locally-flat" projection**

```
PROJCS["SVY21 / Singapore TM",
  GEOGCS["SVY21",
    DATUM["SVY21",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6757"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4757"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",1.366666666666667],
  PARAMETER["central_meridian",103.8333333333333],
  PARAMETER["scale_factor",1],
  PARAMETER["false_easting",28001.642],
  PARAMETER["false_northing",38744.572],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AUTHORITY["EPSG","3414"]]
```

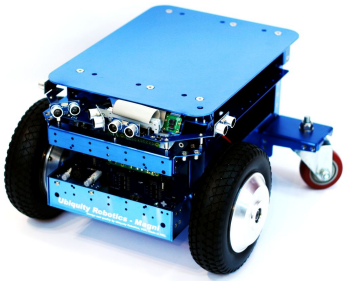
Free Fleet

https://github.com/open-rmf/free_fleet

Free Fleet



- Mobile robot deployments often involve highly customized tasks that commercial fleets do not support, for example interacting with manipulators, custom sensors, etc.
- These tasks are performed by custom mobile robot fleets, that are either built in-house or standalone research platforms. Hence, these mobile robots do not come with their own fleet management system or integration API.
- Free Fleet was introduced to facilitate these custom integrations with RMF, allowing custom fleets to operate in the same space as other RMF-integrated fleets.

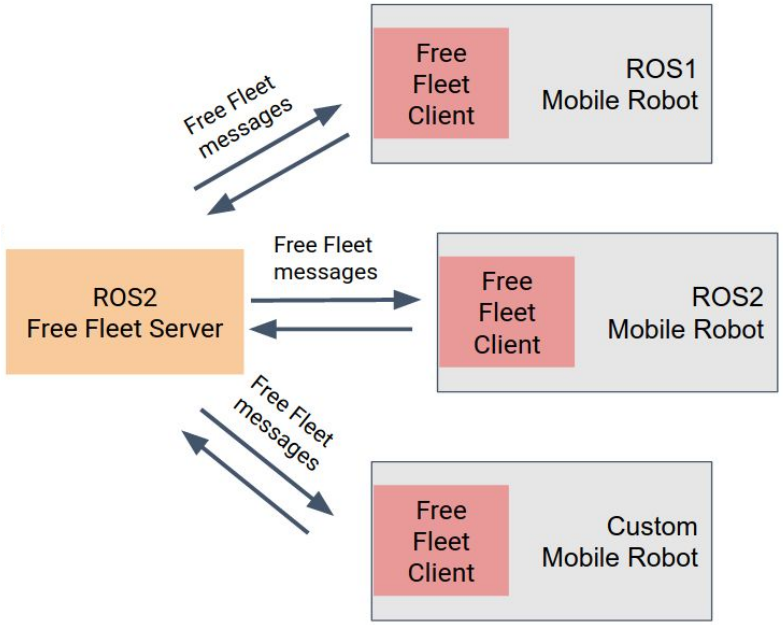


https://github.com/open-rmf/free_fleet

https://github.com/eclipse-cyclonedds/cyclonedds

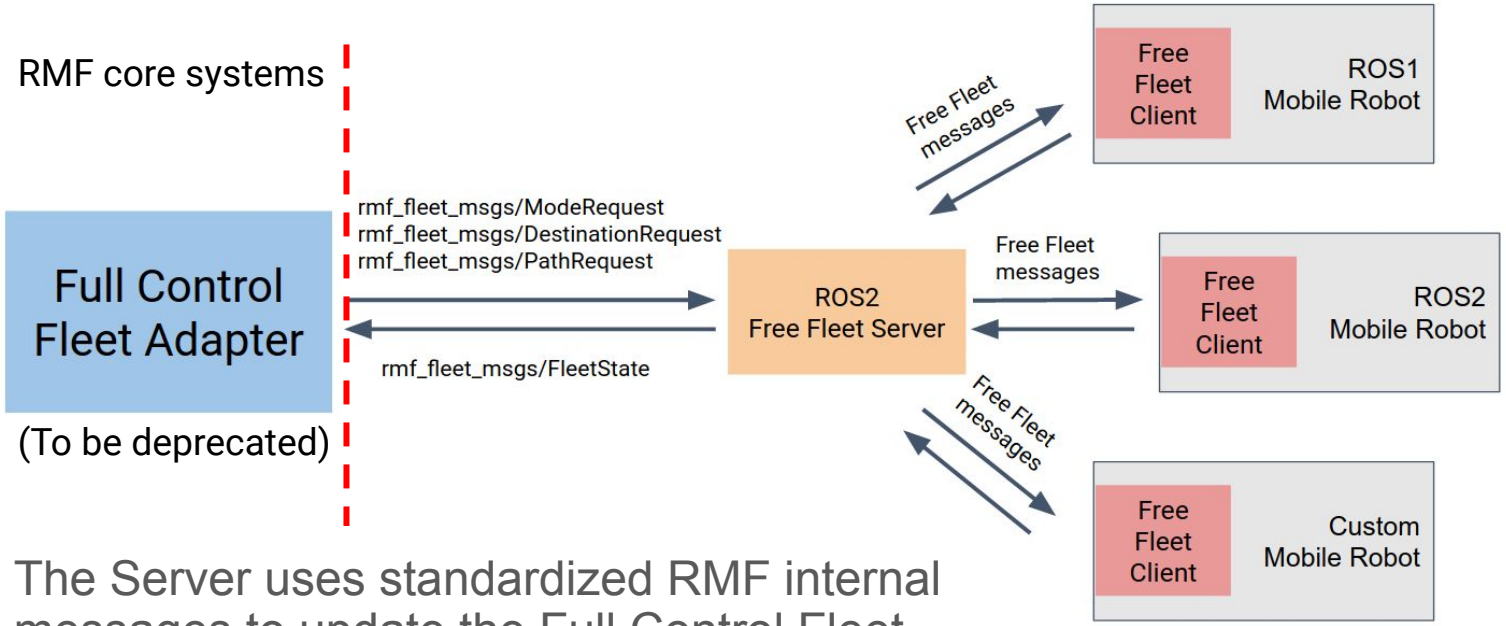
Free Fleet: Overview

- Free Fleet can be divided into 2 main components, the Server and the Client.
- The Clients utilize the commonly used navigation stack API in both ROS 1 and ROS 2, to monitor and command the mobile robot.
- Communication between the Server and Client is achieved using CycloneDDS.



https://github.com/open-rmf/free_fleet

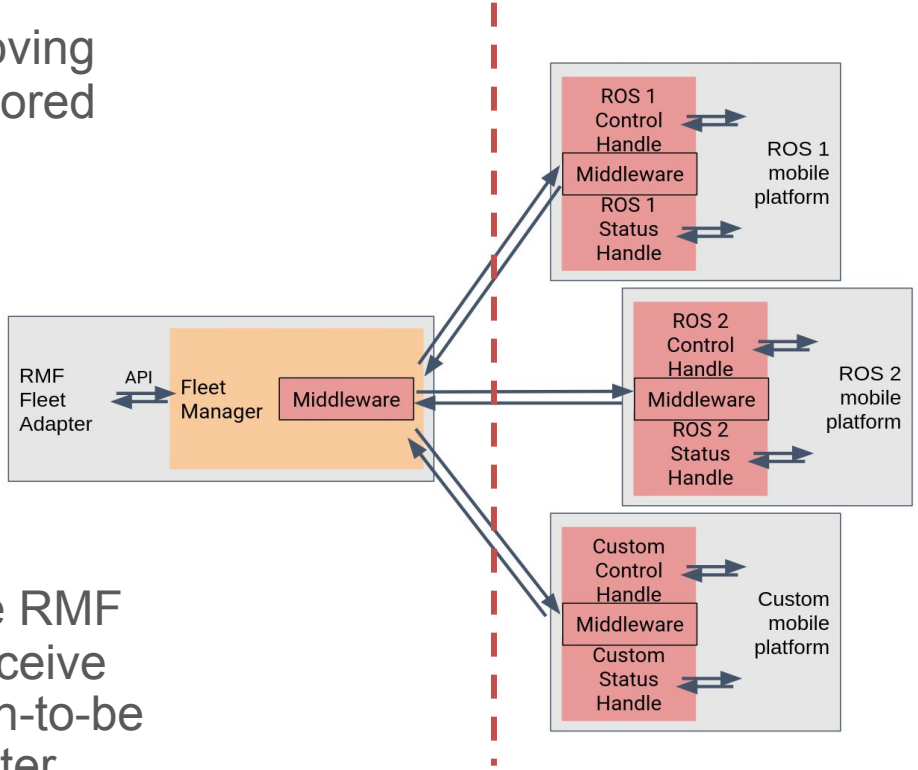
Free Fleet: Current State Overview



The Server uses standardized RMF internal messages to update the Full Control Fleet Adapter, as well as provide RMF control over the client robots.

Free Fleet: Moving forwards

- To increase user configurability, moving forwards, Free Fleet is being refactored into a collection of libraries and interfaces, instead of executables.
- This facilitates integration with custom
  - Navigation stacks
  - Middlewares
  - Fleet management interfaces
- The fleet manager will then use the RMF Fleet Adapter API to update and receive commands, as opposed to the soon-to-be deprecated Full Control Fleet Adapter.



Adapters & Templates

https://osrf.github.io/ros2multirobotbook/integration.html

Adapters & Templates Overview

Robot Fleets

Doors

RMF uses ROS 2 messages and topic interfaces to communicate between different components in the overall RMF system. In most cases we use components called Adapters to bridge between the hardware-specific interfaces and the general purpose interfaces of RMF

Workcells

Lifts

Enterprise Systems

User Interfaces

https://osrf.github.io/ros2multirobotbook/rmf-core.html#fleet-adapters

Mobile Robot Fleet Adapter Overview

Current state includes 4 broad categories of fleets

	Full Control	Traffic Light	Read Only	No Interface
Compatible				
API Available				
Provide Regular Status Update				
Provide Live Status Update				
Allow Pause/Resume				
Allow Path Level Control				

- The facility owner and SI will need to decide which level is required
- An advanced integration fleet adapter is planned for future development

Mobile Robot Full Control Fleet Adapter

Assumptions:

- mobile robot fleet manager allows explicit paths to be specified
- the path can be interrupted at any time and replaced with a new path
- each robot's position will be updated live as the robots are moving

API has 4 critical classes:

- Adapter - Initializes and maintains communications with RMF. Register one or more fleets and receive a FleetUpdateHandle for each fleet.
- FleetUpdateHandle - Allows user to configure a fleet by adding robots and specifying settings for the fleet (e.g. specifying what types of deliveries the fleet can perform). New robots can be added to the fleet at any time.
- RobotUpdateHandle - Use this to update the position of a robot and to notify the adapter if the robot's progress gets interrupted.
- RobotCommandHandle - This is a pure abstract interface class. The functions of this class must be implemented to call upon the API of the specific fleet manager that is being adapted.

Fleet Adapter Development - Helpful Resources

Fleet Adapter Overview: https://osrf.github.io/ros2multirobotbook/rmf-core.html#fleet-adapters

Navigation Maps: https://osrf.github.io/ros2multirobotbook/integration_nav-maps.html

Full Control Fleet Adapter template: https://github.com/open-rmf/fleet_adapter_template

Full Control MiR 100 Fleet Adapter template: https://github.com/osrf/fleet_adapter_mir

Full Control API: https://github.com/open-rmf/rmf_ros2/tree/main/rmf_fleet_adapter

Traffic Light API:

https://github.com/open-rmf/rmf_ros2/blob/main/rmf_fleet_adapter/include/rmf_fleet_adapter/agv/EasyTrafficLight.hpp

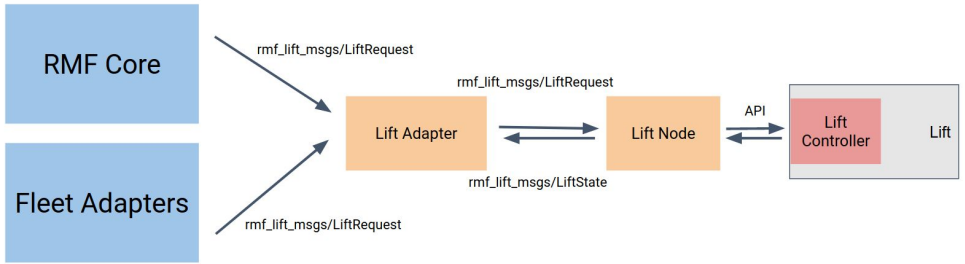
Read Only API:

https://github.com/open-rmf/rmf_ros2/blob/main/rmf_fleet_adapter/src/read_only_blockade/FleetAdapterNode.cpp

Lift (Elevator) Adapters

Multi-story robot operations are possible with RMF resolving conflicts and managing shared resources on a larger scale.

The basic requirement is that the lift controller accepts commands using a prescribed protocol (i.e. OPC, REST, etc.)



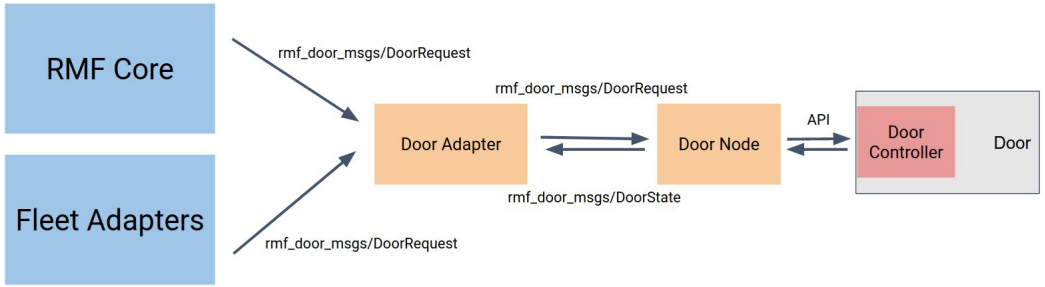
Note that a simple hardware integration is typically required to interface with the lift controller.

Lift Node example: https://github.com/sharp-rmf/kone_lift_controller

https://osrf.github.io/ros2multirobotbook/integration_doors.html

Door Adapters

Doors can be integrated with RMF using a ROS 2 door node and a door adapter, which we sometimes refer to as a door supervisor.



Note that a simple hardware integration is typically required to interface with the door controller.

Door Adapter Template: https://github.com/open-rmf/door_adapter_template

https://osrf.github.io/ros2multirobotbook/integration_workcells.html

Workcell Adapters

Currently RMF has 2 types of sample workcells, namely: Dispenser and Ingestor.

Message Types	ROS2 Topic	Description
rmf_dispenser_msgs/DispenserRequest	/dispenser_reqeusts	Direct requests subscribed by the dispenser node
rmf_dispenser_msgs/DispenserResult	/dispenser_results	Result of a dispenser request, published by the dispenser
rmf_dispenser_msgs/DispenserState	/dispenser_states	State of the dispenser published by the dispenser periodically
rmf_ingestor_msgs/IngestorRequest	/ingestor_requests	Direct requests subscribed by the ingestor node
rmf_ingestor_msgs/IngestorResult	/ingestor_results	Result of a ingestor request, published by the ingestor
rmf_ingestor_msgs/IngestorState	/ingestor_states	State of the dispenser published by the ingestor periodically

Workcell Adapter Nodes:
https://github.com/open-rmf/rmf_simulation/tree/main/rmf_robot_sim_gazebo_plugins/src

Flexible Task System / Web API's

https://github.com/open-rmf/rmf_api_msgs/

Flexible Task System / Web API's

- Defining purpose-built flexible JSON messages instead of directly copying around static ROS messages
  - Static ROS messages cannot be extended into new data structures without negatively impacting the API of the whole system
  - JSON messages can be designed with explicit extension points
- This more flexible API allows us to design RMF to support any type of task, now and into the future, without pre-existing knowledge of robot capabilities
  - System integrators can define entirely new task types that incorporate their own unique robot capabilities, all without changing anything upstream or downstream in RMF
- It will also allow users and system integrators to create custom task requests on the fly by assembling sequences of robot activities into a novel task definition

https://github.com/open-rmf/rmf_task

Flexible Task System Overview

Legacy Approach

- Rigid task definitions
  - Out of the box support for Clean, Loop and Delivery tasks
  - Supporting a different type of task requires extensive development in
    - rmf_task
    - rmf_internal_msgs (ABI breaks)
    - rmf_fleet_adapter libraries
- Unable to interrupt/resume and cancel on-going tasks
- No backup of task/robot states and logs

RMF 22.02 Release

✓ Flexible task definitions
  - **Tasks definitions can now be constructed at runtime** using building blocks defined by json schemas
  - Out of the box building blocks include
    - GoToPlace
    - WaitFor
    - Pickup
    - Dropoff
    - PerformAction and more…

✓ Able to interrupt and resume ongoing tasks

✓ Able to cancel ongoing tasks and have robots perform specifiable behavior upon cancellation

✓ Backup task/robot states and logs

https://github.com/open-rmf/rmf_api_msgs/

Flexible Task API schema - Task Request

27 lines (27 sloc) | 1015 Bytes — Raw | Blame

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://raw.githubusercontent.com/open-rmf/rmf_api_msgs/main/rmf_api_msgs/schemas/task_request.json",
  "title": "Task Request",
  "description": "Describe a task request",
  "type": "object",
  "properties": {
    "unix_millis_earliest_start_time": {
      "description": "(Optional) The earliest time that this task may start",
      "type": "integer"
    },
    "priority": {
      "description": "(Optional) The priority of this task. This must match a priority schema supported by a fleet.",
      "type": "object"
    },
    "category": { "type": "string" },
    "description": {
      "description": "A description of the task. This must match a schema supported by a fleet for the category of this task request."
    },
    "labels": {
      "description": "Labels to describe the purpose of the task dispatch request",
      "type": "array",
      "items": { "type": "string" }
    }
  },
  "required": ["category", "description"]
}
```

https://github.com/open-rmf/rmf_api_msgs/

Flexible Task System - example patrol request

Legacy

task description in ros2 msg

```
rmf_task_msgs.msg.TaskDescription(
        start_time=builtin_interfaces.msg.Time(sec=0, nanosec=0),
        priority=rmf_task_msgs.msg.Priority(value=0),
        task_type=rmf_task_msgs.msg.TaskType(type=1),
        station=rmf_task_msgs.msg.Station(
                task_id='',
                …),
        loop=rmf_task_msgs.msg.Loop(
                 task_id='',
                 robot_type='',
                 num_loops=3,
                 start_name='coe',
                 finish_name='lounge'),
        delivery=rmf_task_msgs.msg.Delivery(
                 task_id='',
                 items=[],
                 pickup_place_name='',
                 ...),
        clean=rmf_task_msgs.msg.Clean(start_waypoint=''))
)
```

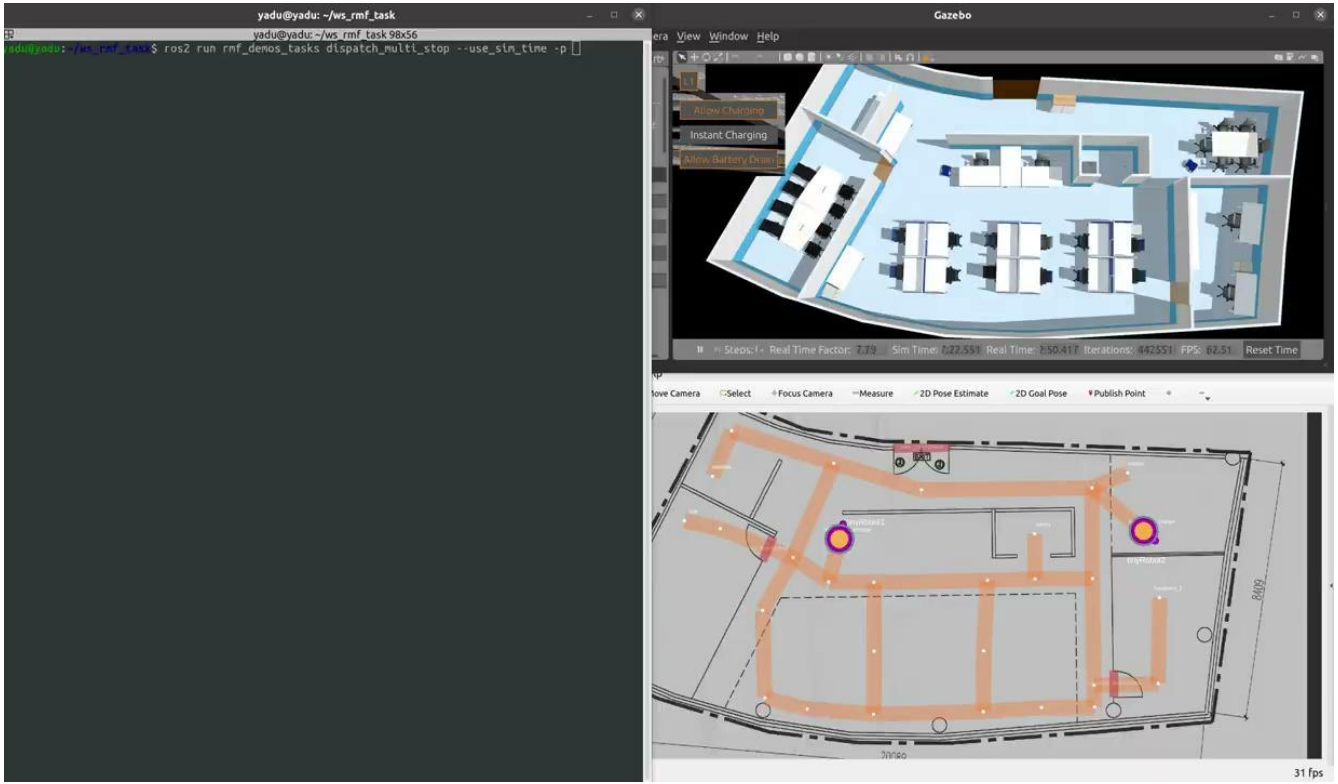
RMF 22.02 Release

task description in json format

```
{
 "type": "dispatch_task_request",
 "request":
 {
  "unix_millis_earliest_start_time": 0,
  "priority": {"type":"binary","value":0},
  "labels": ["swagger"],
  "category": "patrol",
  "description":
   {
    "places": ["coe", "lounge"],
    "rounds": 3
   }
 }
}
```

https://github.com/open-rmf/rmf_task

Flexible Task System Demo: Multi-stop Task

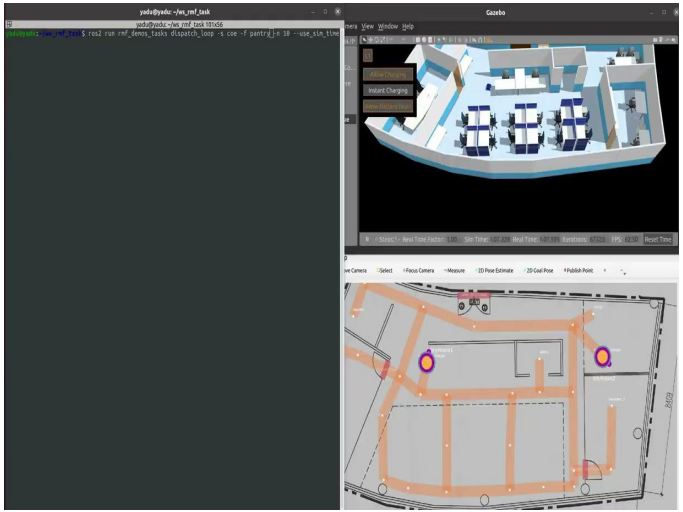


Create and submit a task where a robot needs to visit N different places

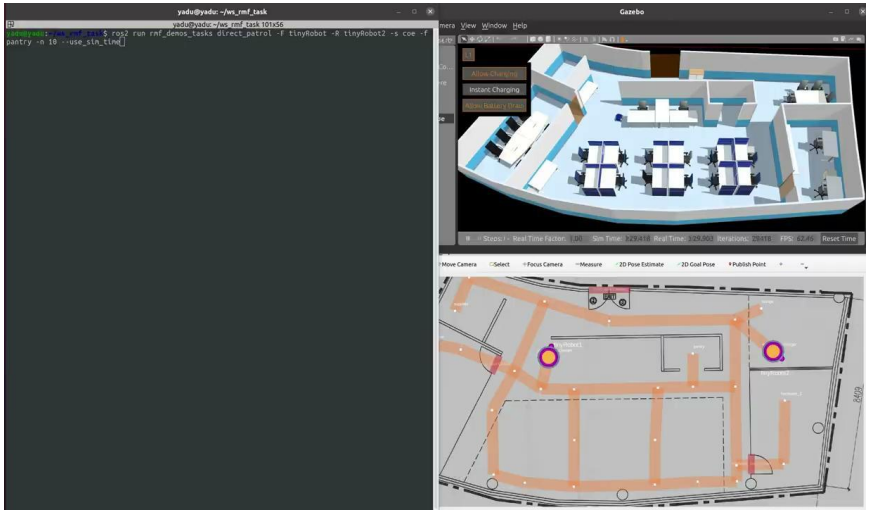
https://github.com/open-rmf/rmf_task

Flexible Task System Demo: Direct Assignment

"Direct Assignment" pipeline implemented in parallel to the bidding framework



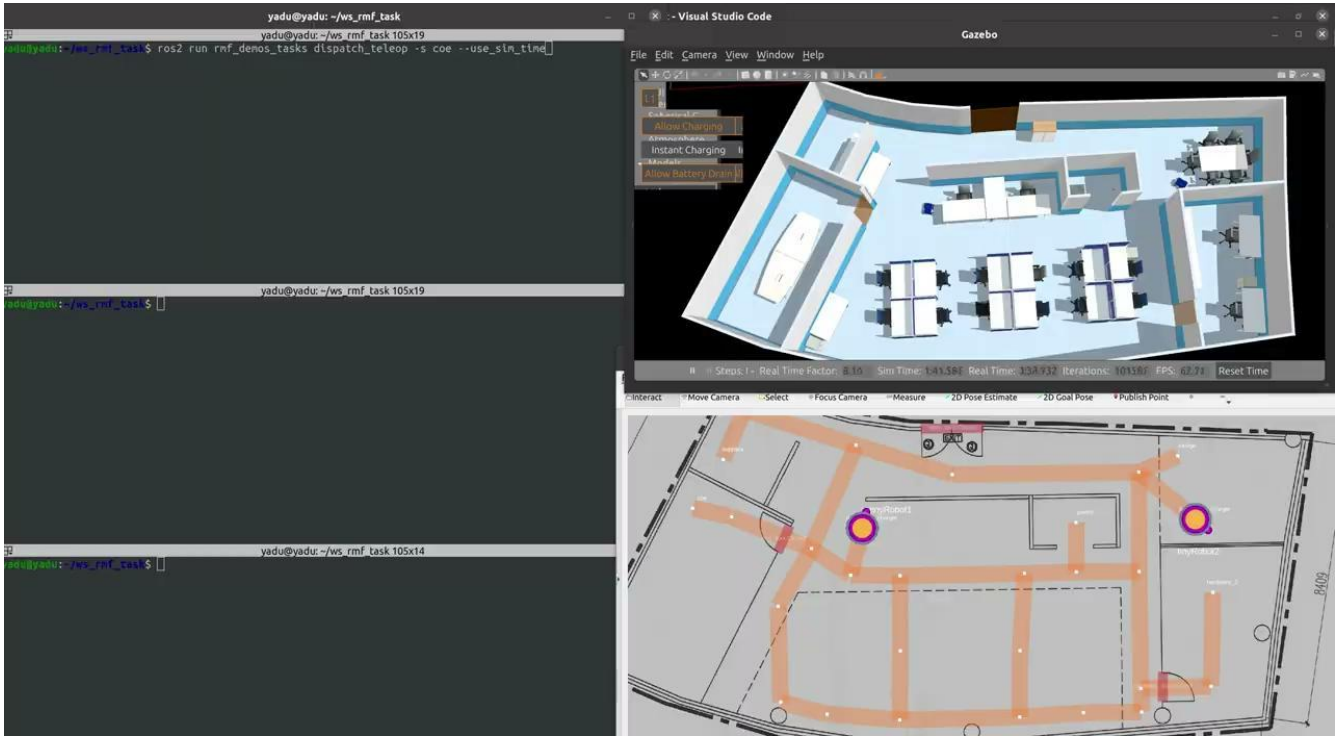
Task is assigned to a robot **tinyRobot1** based on bidding outcome.



Same task is assigned to the **specified tinyRobot2** when submitted via the direct assignment pipeline

https://github.com/open-rmf/rmf_task

Flexible Task System Demo: Teleop Task



- RMF guides robot to teleop start point and then gives up control of the robot (yellow marker disappears)
- Second terminal teleops the robot around (replacement for joystick)
- When done, RMF takes back control (yellow marker appears).

Teleoperation

https://github.com/open-rmf/rmf-web/tree/proto/teleop

Open-RMF Teleoperation

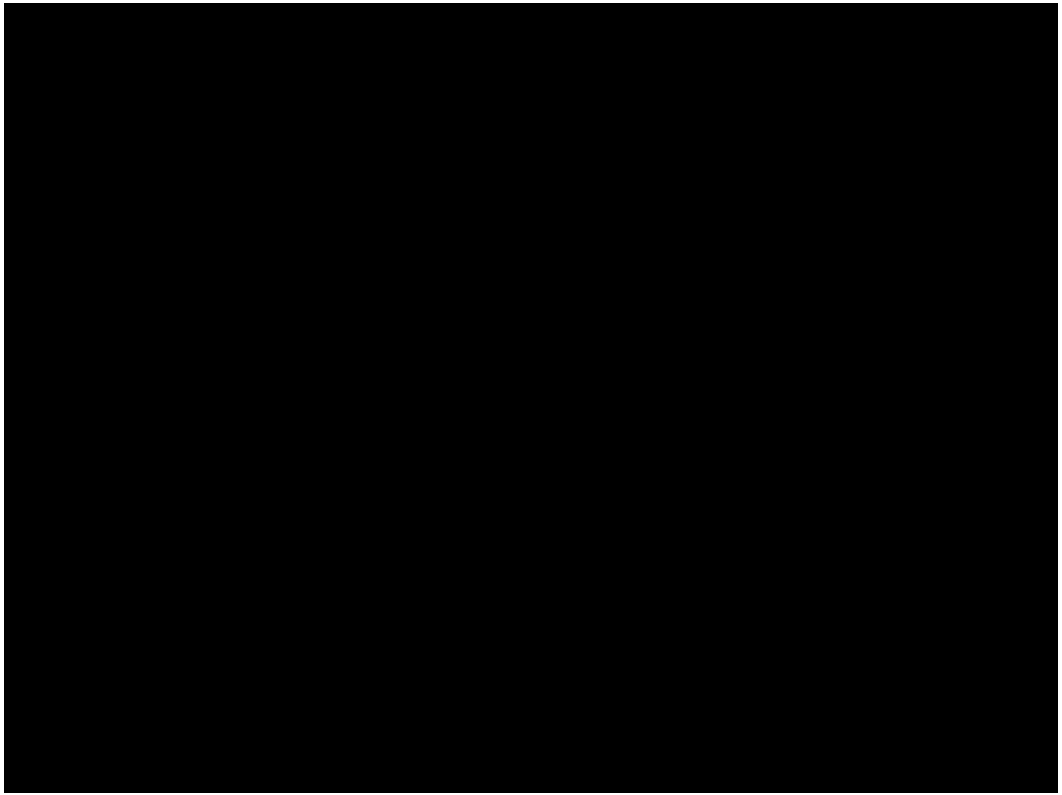
Motivations:

- Open-RMF web dashboard as an operator interface
  - What if Operator wants to override RMF in some cases
  - Possible scenarios
    - Unexpected obstacles / Robot gets “lost” - Robot Navigation Failure
    - “On-Demand” behaviors required - Bypass RMF automation
  - **(A)** Provide standard API mechanisms to enable “common” robot control actions
    - Drive / Reverse / Rotate with Joystick Controls
    - Video Feed of Robot perspective / Navigation map
  - **(B)** Allow mechanisms to flexibly enable robot-unique remote control functions
    - Eg: Tilt of Camera View, Custom speech etc

https://github.com/open-rmf/rmf-web/tree/taskv2-teleop-monolith

Open-RMF Teleoperation

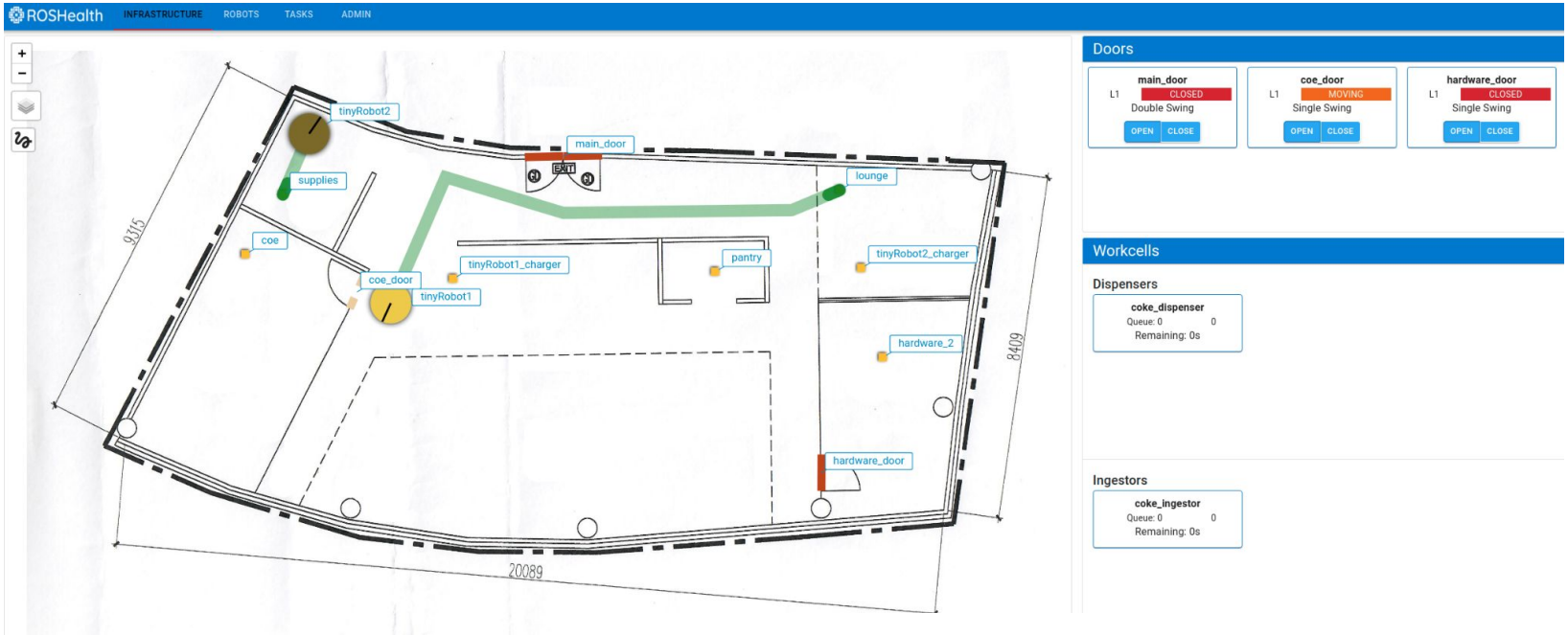
- RMF web server exposes REST / socketio endpoints for generic motion commands
- Adapter translates these messages into robot specific teleoperation control commands
- With Taskv2 PerformAction, Operator temporarily suspends RMF control
- Robot goes vroom!



RMF Web

https://github.com/open-rmf/rmf-web

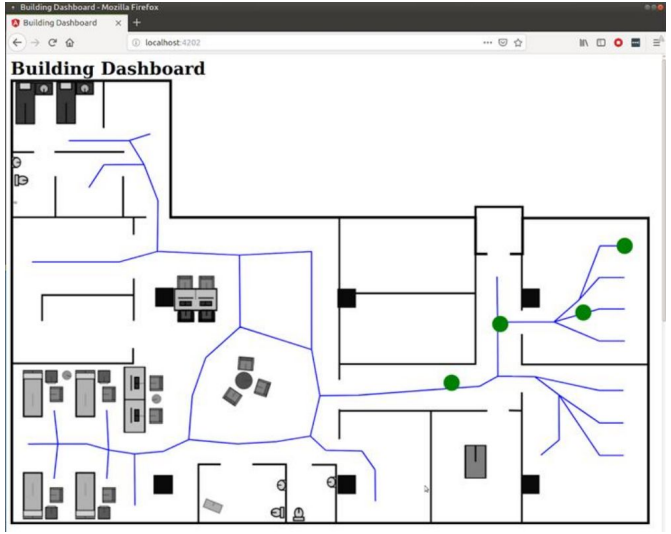
RMF Web: Dashboard



The RMF Web Dashboard is an Open Source, configurable web application that provides users with high level control and visibility over RMF deployments.

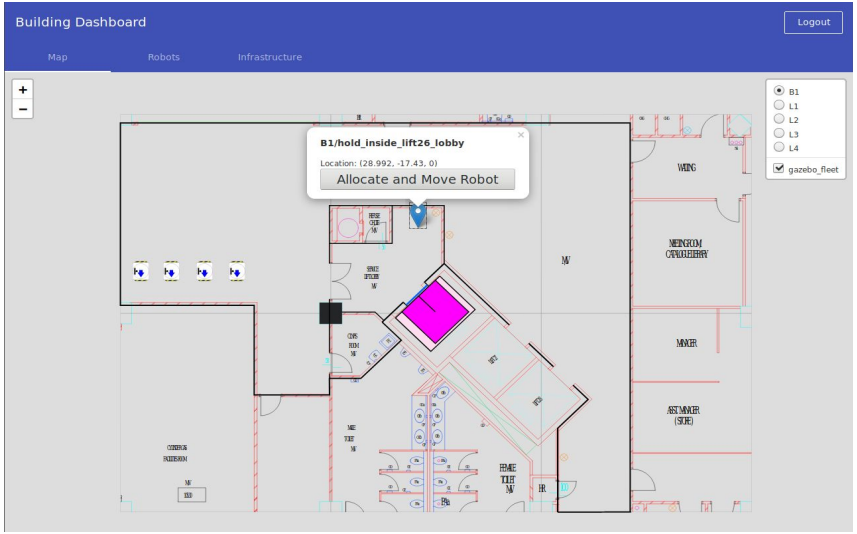
https://github.com/open-rmf/rmf-web

RMF Web: Dashboard evolution



2018

- Written in AngularJs (https://angularjs.org/)
- SOSS ros2-websocket bridge
- Visualization of fleet states
- Barebones prototype



2019

- Written in AngularJs (https://angularjs.org/)
- Visualization of infrastructure states
- Navigation command of robots

https://github.com/open-rmf/rmf-web

RMF Web: Evolution continued

2020

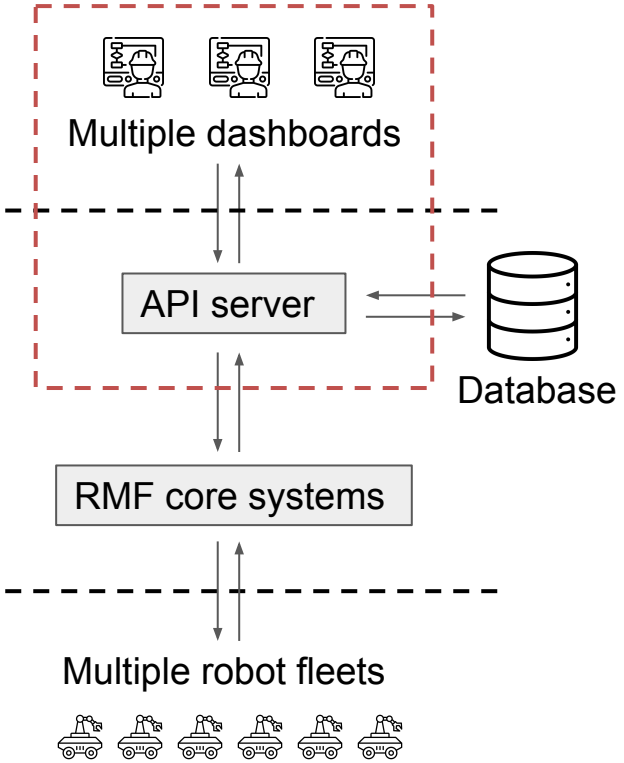
- Migrated to ReactJs (https://reactjs.org/)
- Detailed introspection of states
- Supported Lift and Door control
- Supported Loop and Delivery requests

2022

- Migrated to python backend (api-server) (https://github.com/open-rmf/rmf-web/tree/main/packages/api-server)
- Deprecated use of ros2-websocket bridge
- Separate tabs for Infrastructure, Robots, Tasks and Admin
- Granular introspection of Tasks, Phases and Events, including state and logs

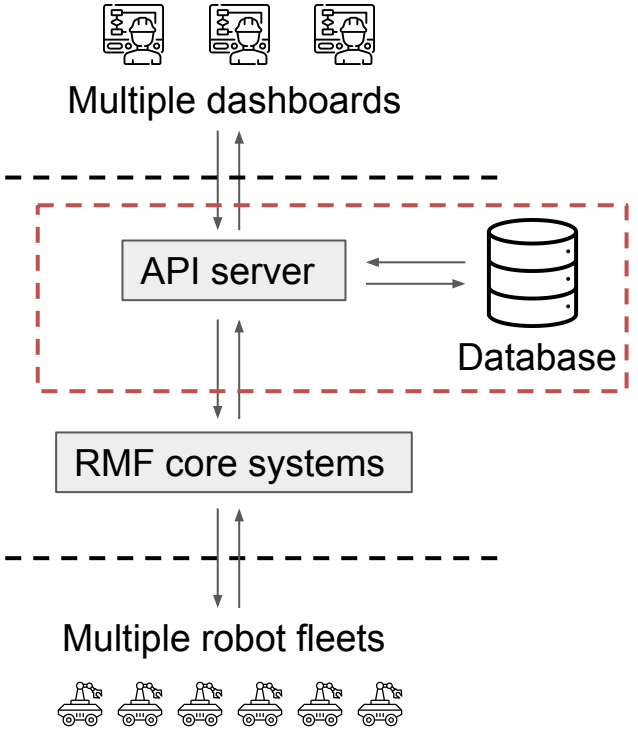
RMF Web: Overview

- The RMF Web stack mainly consists of the API server and the Dashboard.
- The API server implements REST APIs using Socket.IO (https://socket.io/), for all data enquiries from the Dashboards.
- The Dashboard communicates with the API server using OpenAPI (https://swagger.io/specification/) generated API clients.
- The frontend of the Dashboard is implemented using the React framework (https://reactjs.org/).



RMF Web: Overview

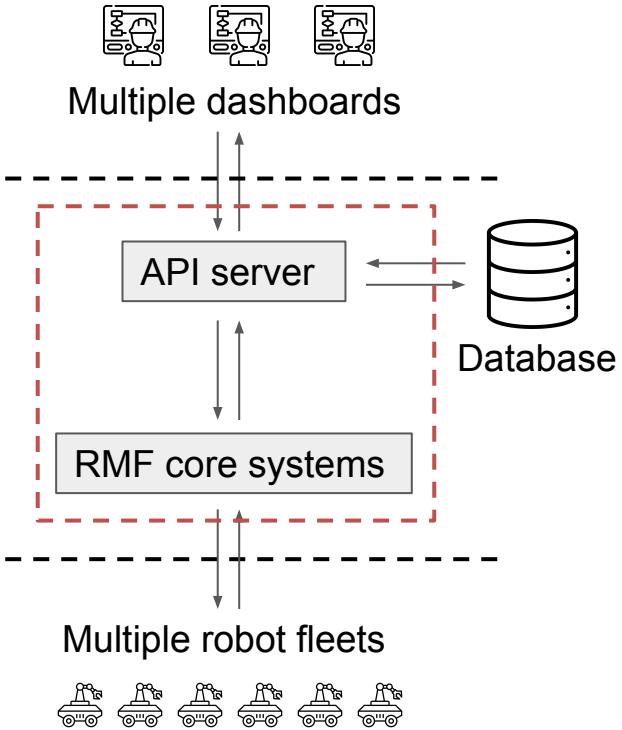
- In order to support multiple types of databases, the API server uses Tortoise ORM (https://tortoise-orm.readthedocs.io/en/latest/) to perform all database related operations, this includes logging and retrieving data.
- Supported databases include
  - PostgreSQL
  - SQLite
  - MySQL / MariaDB



https://github.com/open-rmf/rmf_internal_msgs
https://github.com/open-rmf/rmf_api_msgs

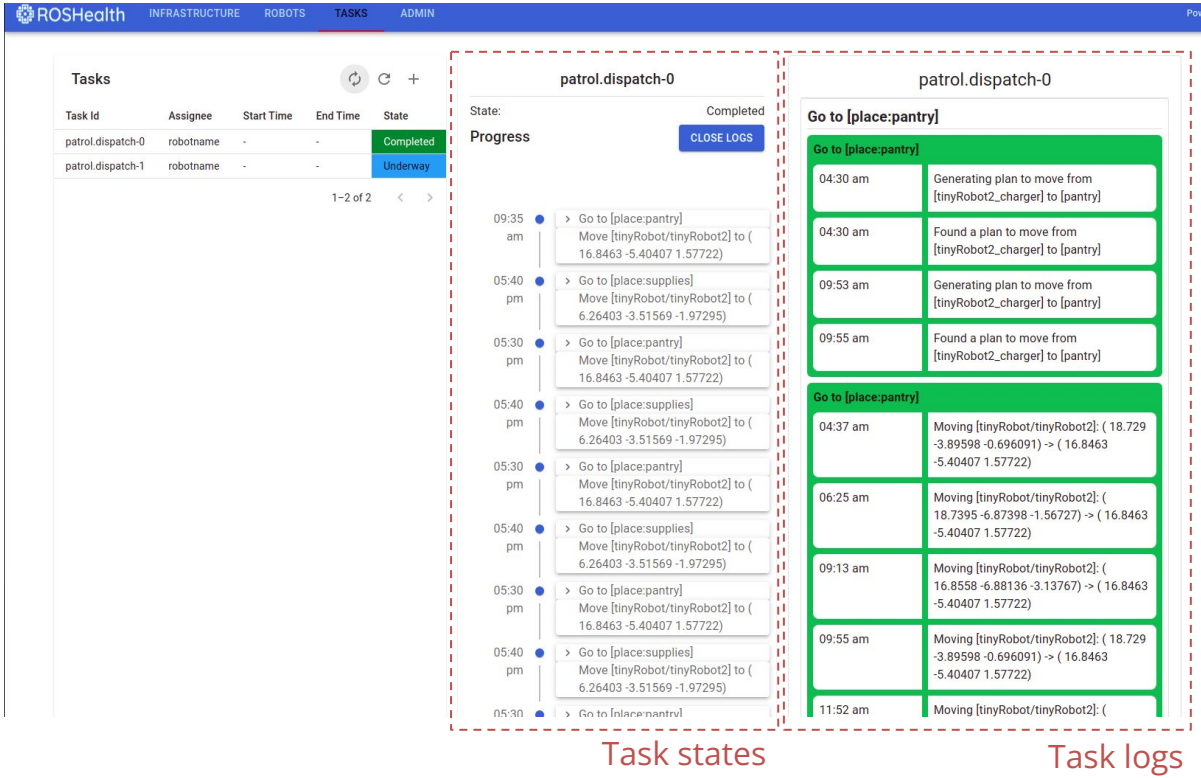
RMF Web: Overview

- The API Server communicates with RMF through
  - Standardized internal messages for anything related to
    - Doors
    - Lifts
    - Workcells
  - Standardized RMF API messages for anything related to
    - Tasks
    - Robot fleets



https://github.com/open-rmf/rmf-web

RMF Web: recent Dashboard upgrades



- rmf-web now provides detailed visualization of Task states and Task logs
- Task states can be expanded into its various phases and events.
- Task logs are described for each separate event available in this task.

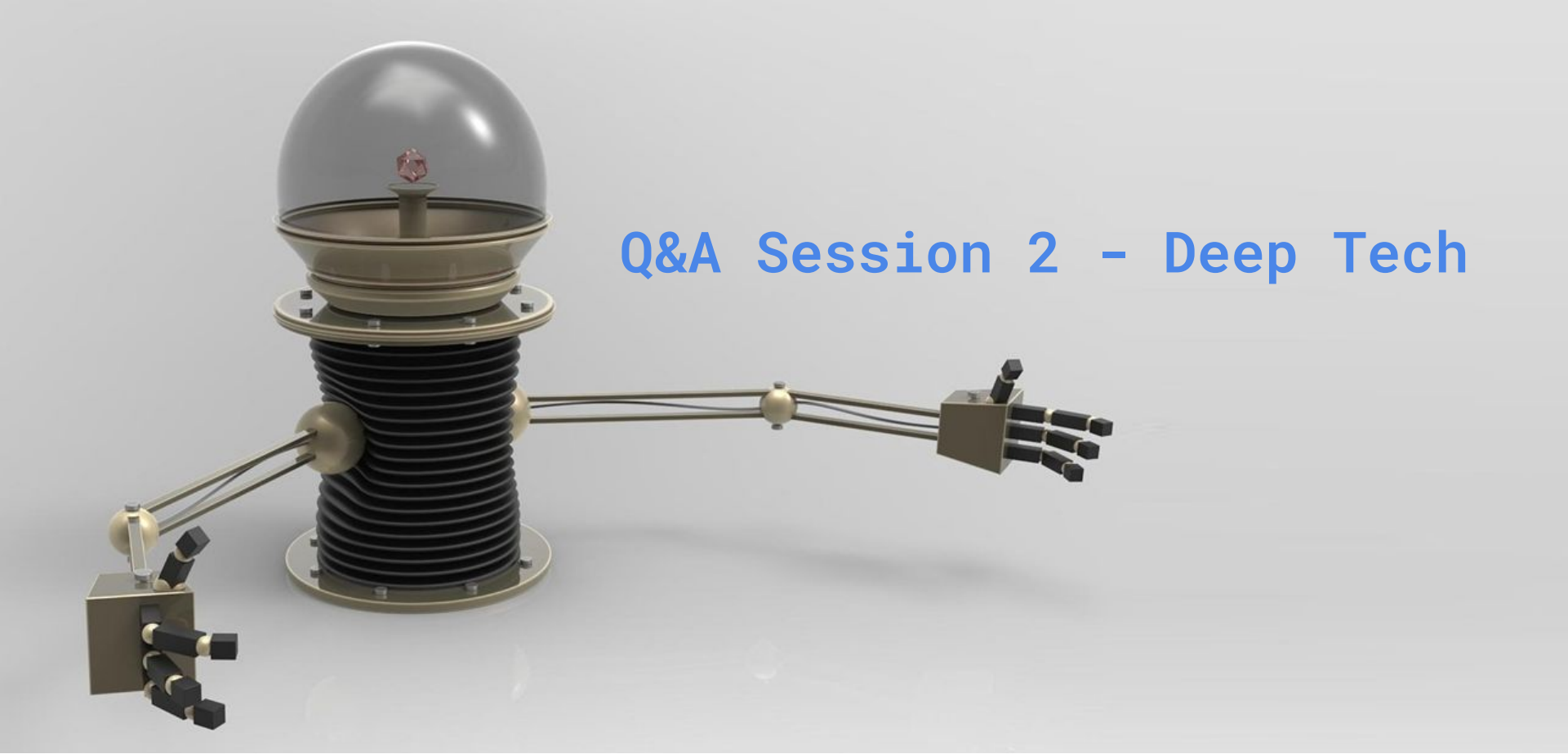
Release Review

Current Release Features

- Infrastructure Integration for Doors, Lifts and Workcells
  - Tested doors - Assa Abloy, Dormakaba
  - Tested lifts - Kone, Meyer, Fujitec
  - state/request/results messages
- Different level of traffic control for robots *(Full Control, Traffic Light, Read Only)*
- Cloud deployment on AWS EC2 with Wireguard VPN
- Web Based GUI Traffic Editor for multi robot deployment
  - Task management and
  - Path planning and editing
  - Infrastructure status
- Traffic scheduler and multi-robot deconfliction capabilities
- Ignition and Gazebo simulator plugins
- Cross-Platform, Cross-Fleet and Cross-Vendor robot interoperability

22.02 Release Review

- Real-Time Location Service (RTLS) Messages
  - Used to track the locations of non-robot assets in real time
- Flexible task definitions
  - System integrators can define new types of tasks without modifying RMF source code
  - The dashboard could expose a UI that lets end-users graphically define custom tasks
    - The end customer of a deployment can decide whether this feature should be made available to users
- Manual intervention of task flow
  - Command a robot to pause or cancel a live task
  - Command a robot to repeat or skip some phase(s) of a task
- Formal robot command hand-off
  - Hand off command of a robot as a formal step in a task
  - Some tasks require domain-specific behaviors for periods of time. This feature provides a formal, seamless way to allow that during ordinary task execution.
- Direct robot task requests
  - Bypass the dispatch system and issue task requests directly to specific robots
- Stable Web API
  - We will be defining a stable Web API for external applications to tap into RMF
  - The API can be used by web dashboards or any other peripheral RMF applications
  - The API will be based on JSON and will transmit using WebSockets and/or REST services depending on API calls



Q&A Session 2 - Deep Tech

Centre for Healthcare Assistive & Robotics Technology

IHiS Because IT's in Healthcare

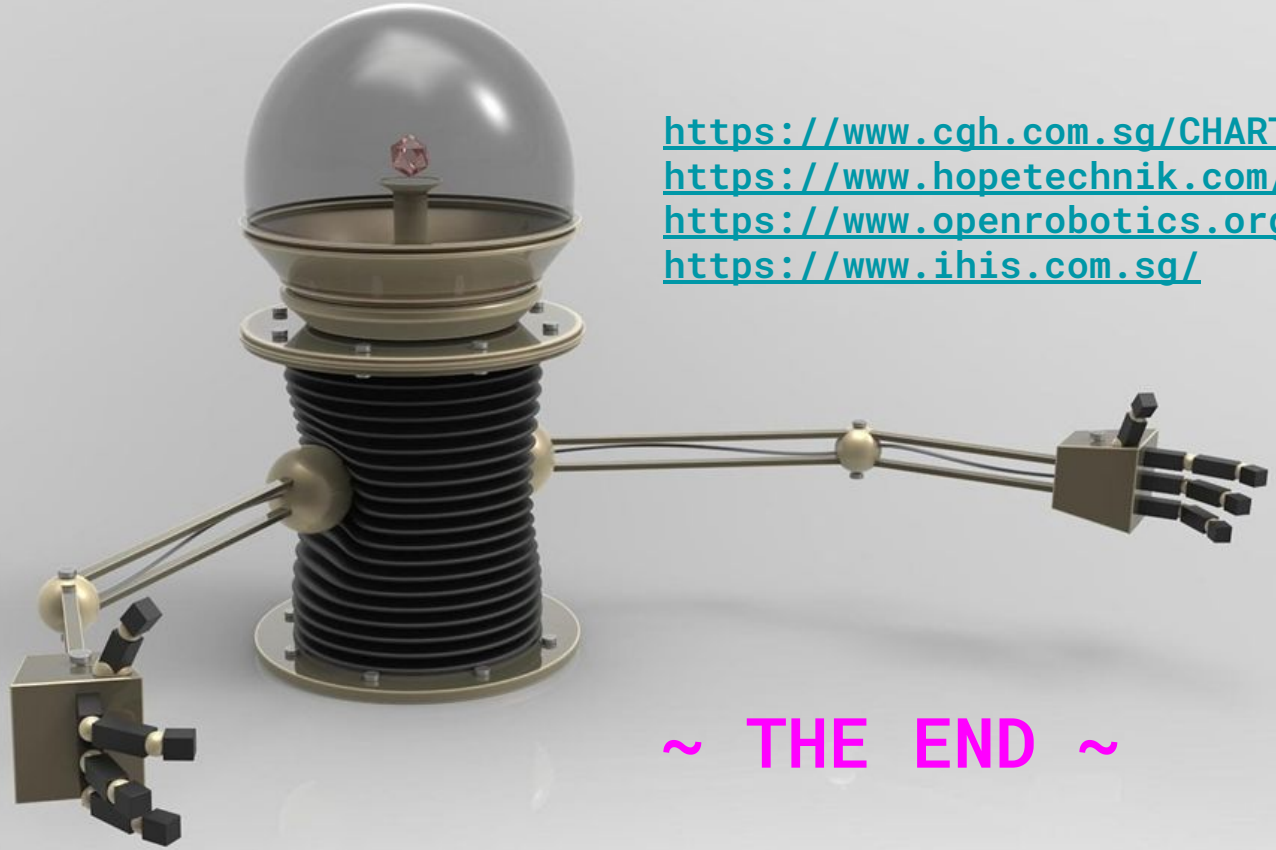
HOPE technik

open robotics

Catalysed by

MINISTRY OF HEALTH SINGAPORE

NATIONAL ROBOTICS PROGRAMME



https://www.cgh.com.sg/CHART
https://www.hopetechnik.com/
https://www.openrobotics.org/
https://www.ihis.com.sg/

~ THE END ~

Centre for Healthcare Assistive & Robotics Technology
CHART
IHiS
Because IT's in Healthcare
HOPE technik
open robotics
Catalysed by
MINISTRY OF HEALTH SINGAPORE
NATIONAL ROBOTICS PROGRAMME