Robotics Middleware for Healthcare (RoMi-H)

Introduction

Robotics Middleware for Healthcare (RoMi-H) is a middleware that aims to integrate diverse systems in the Singapore Public Health Institutions (PHIs). These systems include medical devices, robotic systems, nurse/patient/operator user interface devices (UI), building infrastructure, IoT devices, and Hospital Information Systems (HIS). Refer to Figure 1 for an illustration of RoMi-H.

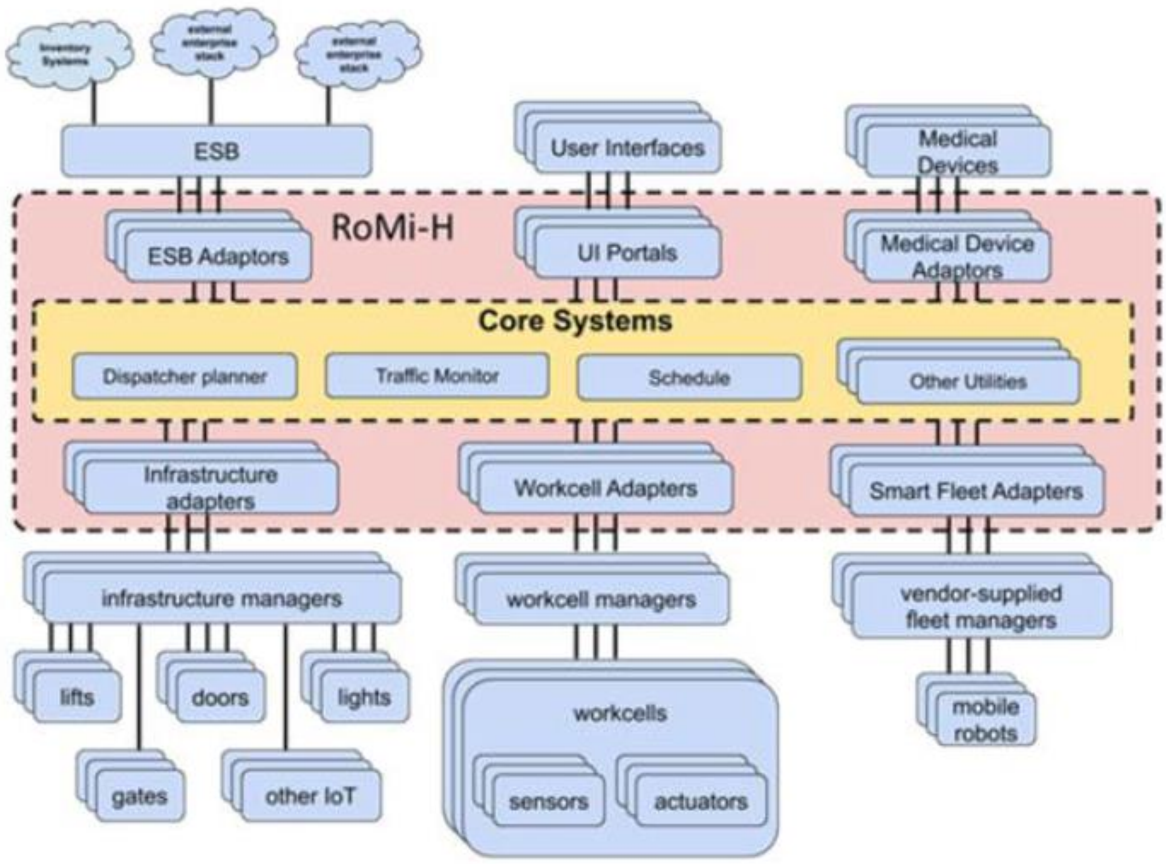


Figure 1 Illustration of RoMi-H

Built on ROS 2 framework, RoMi-H forms the central communication layer that facilitates diverse systems to work on one unified platform, while complying with healthcare standards and controls. RoMi-H achieves this by creating standardised message libraries, translators and adaptors for various platforms.

The interoperability among the heterogeneous systems will allow hospitals to maximise their operational efficiency, such as flexible job allocation to Autonomous Mobile Robots (AMRs) from different vendors. Building infrastructure can also be better utilised by different AMRs e.g. sharing the same lift by different make/model of AMRs. For hospitals, this will mean lower integration efforts for subsequent introduction of new systems, as they only need to integrate their common systems and infrastructure once via RoMi-H.

RoMi-H will also enable adoption of standardised security protocols, reduce technical risks, and allow the healthcare industry to deploy solutions more quickly to satisfy the growing demand for automation and adoption of technology.

Key Features

- **Infrastructure management**: RoMi-H allows multiple robotic agents to communicate with building infrastructure, building management systems and enterprise systems (e.g. card access control, door and lift controls, fire safety systems and nurse call systems).
- **Multi-fleet coordination**: RoMi-H Core provides a scalable, highly modular and distributed management system to coordinate and monitor multiple fleet managers, building infrastructure and work cells.
- **Simulation**: RoMi-H comes with a virtual simulation tool to help Institutions identify and predict optimal traffic by modelling the deployment of robots in the Institution. This tool will also help in expansion planning and real time code testing of the systems.
- **Development and integration tools**: Using ROS 2 provides access to an evergrowing community and open-source packages. Some recent developments include:
    - System-Of-Systems Synthesizer (SOSS) – a bridge allowing the conversion of messages between different protocols, including but not limited to ROS 2, ROS1, WebSocket, TCP server, TCP client, REST server and HL7.
    - Robot Agnostic Mapping Platform (RAMP) – a mapping platform for generating 3D maps of the environment. The maps can be used by 3D LiDAR-based robots or be sliced to fit any 2D LiDAR-based robot.
    - ROS 2 packages for medical grade devices, LiDARs, location beacons, preventative maintenance sensors and more.
    - Health IT (HIT) sandbox for development and integration environments.

RAISE and Robot Integration

The Robotics, Automation, and Interfaces Scheduling Engine (RAISE) is designed as a high-level planner and operations facilitator for robotic task assignment and execution. This system will work independently of and in parallel with vendor fleet managers. The RAISE does not replace vendor fleet managers; however, vendor fleet managers will be required to interact with the RAISE through a set of APIs in order for the building's robotic ecosystem to function properly.

The RAISE will track, communicate and monitor the status of robots as well as their task assignments. Priority levels will be assigned to tasks within the RAISE in order to ensure higher priority plans will take precedence. For example, an urgent pharmaceutical ad hoc delivery will override an ad hoc beverage delivery.

In addition to task assignment and execution management, the RAISE will need to arbitrate robot traffic flows and manage infrastructure (i.e. doors, lifts, etc) in order to optimize the ecosystem. This arbitration logic is required to:

1. Avoid potential conflicts between platforms of different vendors and
2. Resolve conflicts should they arise.

More technical details of Robotic Middleware Framework and available packages are available at: https://osrf.github.io/ros2multirobotbook

Appendix 1 – RoMi-H Compliance & Integration Requirements

A. General Requirements

1. A SI will be identified for RoMi-H integration in each institution.
2. The system vendor shall provide the necessary information and commands so as to allow control of the system by RoMi-H in both normal and emergency situations.
3. The connectivity between the vendor system server and RoMi-H shall meet the requirements in accordance to **Error! Reference source not found.** requirements.
4. When called upon by the SI, the system vendor shall provide the necessary information for the verification and validation of RoMi-H software configuration to be installed and run in an institution.
5. All interface messages shall be request / inquiry messages or result / response messages.
6. The system vendor's solution shall only request for messages on a need-to basis.
7. All messages must be buffered until the receiving party confirms the receipt of the message.
8. All messages shall be logged in the system for at least 2 weeks.
9. In the event of lost connection to RoMi-H, the system design shall involve the following procedure:
    i. The sender shall repeat a message until confirmation by receiver or until a pre-defined connection cut-off time.
    ii. Upon cut-off, the system shall send an alert to the SI to alert of issue.
10. The system vendor shall provide the required information to RoMi-H, which may include but not limited to the following (Appendix 3):
    i. Status of AMR which include travel path, updated location, battery life and etc.
    ii. Fault report and rectification report.
    iii. Any other information as requested by SI and agreed by the system vendor.

B. Integration with AMR

1. The AMR solution will be assessed based on the Compliance Levels with RoMi-H. They are explained in Table 1 with the relevant descriptions of each Level.
2. The AMR system shall have a minimum Compliance Level of "Medium" with respect to the information and commands provided to RoMi-H.
3. The fleet manager of the vendor shall communicate to RoMi-H through one of the following means:
    i. For ROS 2 system, the vendor shall provide standardized interface messages as indicated in Appendix 3Appendix 3 – Standardized ROS 2 Interface Messages.
    ii. For Non-ROS 2 system, the vendor shall provide the information and relevant commands of the AMRs.

Table 1: RoMi-H Compliance Levels

Compliance	Expected Data Comms	Outcome
HIGH	• robot locations • read/write goals • pause / resume • read/write path waypoints	RoMi-H able to observe individual AMRs in the fleet; able to direct individual AMRs to specific indoor location
MEDIUM (Minimum Compliance)	• robot locations • read-only goals • pause / resume	RoMi-H able to observe individual AMRs in the fleet; able to pause and resume individual AMRs
LOW	• robot locations • read-only goals	RoMi-H able to observe location of individual AMRs in the fleet; no level of control. Fleet operations must be spatio-temporally separated via manual scheduling or constant manual deconfliction.
NON	• not applicable	Robots are obstacles. Fleet operations must be spatio-temporally separated via manual scheduling or constant manual deconfliction.

4. For Non-ROS 2 system, the information and commands shall be integrated to RoMi-H through REST APIs. The information and commands to integrate are stated in Table 2.

Table 2: AMR Commands

Command/Information	Description
Start	Command from RoMi-H to start moving the AMR
Stop	Command from RoMi-H to stop the AMR
AMR Status	Information to RoMi-H on the battery level, job ID, orientation and speed information of the AMR
AMR Location	Information to RoMi-H on X and Y coordinates of the AMRs on its map
AMR Waypoint (Read)	Information on the X and Y coordinates of the waypoints that the AMR intends to pass through for its current task
AMR Waypoint (Write)	Command from RoMi-H to change the X and Y coordinates that the AMR intends to pass through for its current task
Map	The map used by the AMR on its fleet management system
Floor Change	Command from RoMi-H to change the map of the AMR when it reaches a new/different floor
Request Lift	Command to RoMi-H to request to take lift
Request Lift Destination	Command to RoMi-H to request to send the lift to destination floor
Request Door Open/Close	Command to RoMi-H to request to open or close the door
Request AMR Battery Management System	Command from RoMi-H to AMR fleet manager to allow charging of AMR. This could be in CAN / IO command.

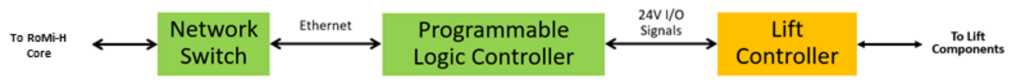
5. The AMR shall update the location and status to RoMi-H on 1 Hz frequency. The frequency could be adjusted by the SI with Institution's consideration.
6. The RoMi-H will provide location and status updates of the AMR via RoMi-H dashboard.

7. The vendor shall work with the SI to ensure that the AMR can execute the following actions during emergencies (e.g. fire / evacuation / Code Blue):
   i. Communicate with the Fire Alarm System/Building Management System via RoMi-H.
   ii. Move to specific, predetermined areas autonomously as commanded by RoMi-H.
   iii. If the AMR is in the lift, it shall exit the lift and move to specific, predetermined areas outside the lift autonomously.

C. Integration with Lift System

1. The vendor shall work with the SI to develop communication protocol that allows the lift system to operate with RoMi-H and the AMR system
2. Possible integration method between lift and RoMi-H include i) Integration via PLC, and ii) Integration via server

i. Lift Integration with PLC

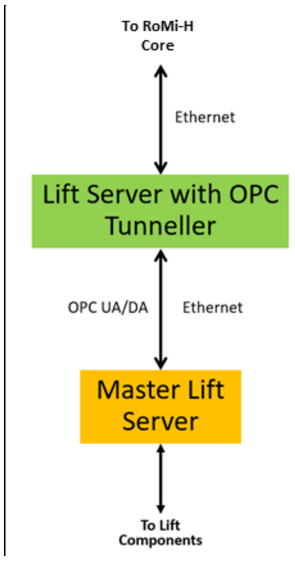


The suggested integration to lifts consists of SI components (green) and the lift OEM components (orange).

In this integration, a PLC takes in, via Ethernet, RoMi-H messages from RoMi-H Core. Pre-programmed logic in the PLC is to recognize and convert these RoMi-H messages to a sequence of 24V input/output signals to the lift controller. These signals may include: door status, lift mode, floor, etc. The sequence of activating the 24V I/O signals must match that from the lift controller in order to control the lift. The SI is to ensure that the OPC variables are properly mapped to RoMi-H messages. This piece of information can be obtained from the OEM of the lift.

Both the SI and the lift OEM must ensure that the lift continues to operate safely after the integration with RoMi-H. Note that AMR movements are not shown here but it is commanded by RoMi-H Core separately.

ii. Lift Integration with Server



Lift integration can also be through server-to-server communications. This happens when the lift OEM also uses a master lift server (orange) to control the lifts.

In this suggested integration, the lift server with OPC tunneller (green) is to be provided by the SI. This server takes in RoMi-H messages from RoMi-H Core; it translates and converts the messages to trigger various OPC variables. The communication with the master lift server is done using an OPC tunneller. The exact OPC variables to trigger depend on the lift OEM. These OPC variables may include: door status, lift mode, floor, etc. The SI is to ensure that the OPC variables are properly mapped to RoMi-H messages.

Both the SI and the lift OEM must ensure that the lift continues to operate safely after the integration with RoMi-H. Note that AMR movements are not shown here but it is commanded by RoMi-H Core separately.

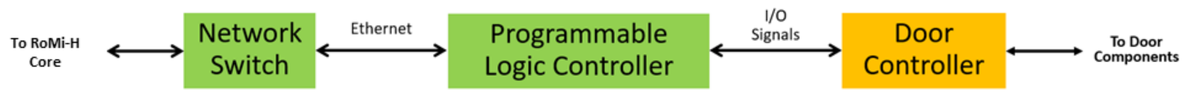
3. With either integration method, the lift system should be able to operate with RoMi-H in the sequence described in Table 3.
4. If the lift system is unable to work with the said sequence, the vendor should work with the SI to create an alternative sequence compliant with RoMi-H architecture.

Table 3: Integration with Lift System

	Sequence	**Signals to/from lift controller (LC) & corresponding actions**
1.	AMR arrives at the way point of the origin lift lobby and stops. AMR informs RoMi-H its location	N.A
2.	RoMi-H makes a request to switch the lift to AGV mode (assuming shared lift)	Signal from RoMi-H to LC that toggles the lift to AGV mode from passenger mode
3.	Lift sends confirmation that lift mode changed to AGV mode	AGV mode Signal from LC to RoMi-H
4.	RoMi-H makes hall call to lift controller to send the lift to the origin floor.	Signal from RoMi-H to LC that corresponds to the hall call level
5.	Lift moves to origin floor and holds its doors open	N.A
6.	Lift controller informs RoMi-H lift has arrived at origin floor and lift doors are open	LC sends floor number and door opened signal to RoMi-H
7.	RoMi-H requests AMR to enter the lift by giving it the next way point inside the lift	N.A
8.	AMR enters the lift and informs RoMi-H it has arrived at the way point inside the lift	N.A
9.	RoMi-H makes car call to lift controller to send the lift to the destination floor	RoMi-H sends LC door close command (if needed) and the destination floor number
10.	Lift controller closes the lift doors, and sends lift to the destination floor	N.A
11.	Lift arrives at destination floor and holds its doors open	N.A
12.	Lift controller informs RoMi-H lift has arrived at destination floor and lift doors are open	LC sends floor number and door opened signal to RoMi-H
13.	RoMi-H requests AMR to exit the lift by giving it the next way point in the lift lobby	N.A
14.	AMR exits the lift and informs RoMi-H it has arrived at the way point in the lift lobby	N.A
15.	RoMi-H informs lift controller to release the lift	RoMi-H sends LC to change the lift mode to passenger mode
16.	Lift controller closes lift doors, switches lift to Passenger Mode, and returns lift to hall call group	N.A
17.	RoMi-H requests AMR to resume mission	N.A
18.	AMR resumes mission	N.A

D. Integration with Door System

1. The vendor shall work with the SI to develop communication protocol that allows the door system to operate with RoMi-H and the AMR system
2. Possible integration method between door and RoMi-H include integration via PLC



The suggested integration to doors consists of SI components (green) and the door OEM components (orange).
In this integration, a PLC takes in, via Ethernet, RoMi-H messages from RoMi-H Core. Pre-programmed logic in the PLC is to recognize and convert these RoMi-H messages to a sequence of 24V input/output signals to the door controller. The signals from the door controller must include the door status i.e. door fully opened or door fully closed. The PLC is to control the opening and closing of doors upon receiving the relevant message from RoMi-H Core.
Both the SI and the door OEM must ensure that the door continues to operate safely after the integration with RoMi-H. Note that AMR movements are not shown here but it is commanded by RoMi-H Core separately.
The SI may propose for integration with the Institution's secure doors system (if any).

3. The door system should be able to operate with RoMi-H in the sequence described in Table 4.
4. If the door system is unable to work with the said sequence, the vendor should work with the SI to create an alternative sequence compliant with RoMi-H architecture.

Table 4: Integration with Door System

	Steps	Signals to/from door controller (DC) & corresponding actions
1.	AMR arrives at the waypoint in front of the door and stops. AMR updates RoMi-H its location	NA
2.	RoMi-H sends 'open door' message to PLC in OPCUA variables. PLC sends 'open door' signal to door controller	- DC receives 'open door' signal from PLC
3.	Door controller opens the door and holds the door open. (*if needed*) PLC sends 'hold door' message to door controller to keep the door open	- DC opens the door - DC holds the door open - *(if needed)* DC receives 'hold door' signal from PLC
4.	Once door is fully opened, door controller sends 'door is open' signal to PLC. PLC sends 'door is open' message to RoMi-H in OPCUA variables	- DC sends 'door is open' signal to PLC
5.	RoMi-H requests AMR to move to the next waypoint after the doorway	NA
6.	AMR moves through the doorway	NA
7.	AMR updates RoMi-H its new waypoint after the doorway	NA
8.	RoMi-H sends 'close door' message to PLC in OPCUA variables. PLC sends 'close door' signal to door controller.	- DC receives 'close door' signal from PLC
9.	Door controller closes the door.	- DC closes the door
10.	Once door is fully closed, door controller sends 'door is closed' signal to PLC. PLC sends 'door is closed' message to RoMi-H in OPCUA variables	- DC sends 'door is closed' signal to PLC
11.	Door returns to normal mode e.g. card access.	NA
12.	RoMi-H requests AMR to resume mission	NA
13.	AMR resumes mission	NA

Appendix 2 – Verification and Validation

Verification and validation post RoMi-H integration in an Institution shall, at a minimum, have testing and simulation standards as defined below.

Testing Standards

Test Type	Requirement	Frequency
Unit Tests	‣ Full coverage of public APIs of all libraries ‣ Coverage of known edge cases in the API (e.g. to verify that exceptions are handled correctly)	Continuous Integration*
Integration Tests	‣ Full coverage of the features provided by the public APIs of all libraries ‣ Coverage of challenging edge cases to ensure desired behavior	Continuous Integration*
System Tests	‣ Full coverage of all executables that are used in RoMi-H ‣ A set of simulated tasks that encompass the critical capabilities of RoMi-H ‣ A set of simulated scenarios that present substantial challenges to RoMi-H (e.g. traffic conflicts in a crowded hallway)	Continuous Integration*
Performance Tests	Testing metrics for: ‣ Latency ‣ Communication bandwidth usage ‣ Quality of deployment (e.g. were the tasks completed in a good time? were resources utilized well?) In scenarios that are: ‣ Easy (e.g. low-traffic) ‣ Nominal (e.g. expected traffic) ‣ Difficult (e.g. high-traffic)	Once per week AND Before deployment
Customer Acceptance Test	‣ Systems Tests that match the customer's use cases ‣ Performance Tests that reflect the customer's infrastructure limits	Before deployment

*Continuous Integration: These tests are run twice each day. These tests must also be satisfied by any new code before merging it into the mainline codebase.

Simulation Standards

Prior to installation, prior to operations and in the event when any new devices are introduced to the system, a software simulation demonstrating (i) virtual physical performance, (ii) successful software command execution (i.e. System Tests as described above), and (iii) performance to Service Level Agreements (SLAs) of the Institution must be performed.

The interfaces to be validated must include all interfaces which will be used in operations of the RoMi-H for that particular Institution. The following are typical interfaces which could be included, however final interfaces must be identified by the integrator and Institution: robot fleet managers, elevators, doors, building management systems, user interfaces, workcells, HIT, IOT sensors and medical devices.

As part of the continuous development of RoMi-H, simulation tools and assets for CHART's Virtual Test-Bedding Platform will be published. It is suggested but not required, that these tools be used for easier evaluation by the approving body.

Appendix 3 – Standardized ROS 2 Interface Messages

To connect to RoMi-H, the vendor shall provide the following ROS 2 messages. Depending on the final use case, the list of ROS 2 message may include more items.

ROS 2 Message	Details (*message type* message name)	Description
FleetState.msg	***string* fleet_name**	Name of the fleet
	***RobotState[]* robots**	Status of robots
RobotState.msg	***builtin_interfaces/Time* robot_time**	Time of robot
	***string* robot_name**	Name of robot
	***string* status**	Status of robot
	***geometry_msgs/Pose* location**	Location of robot
	***Task[]* task_queue**	Task queue
	***float32* battery_percent**	Percentage of battery left

	***uint32* mode**	Mode of robot
		Normal = 0
		Charging = 1
		Pause = 2
		Emergency = 3

RobotTask.msg	***string* robot_name**	Name of robot
	***string* task_id**	ID of task
RobotPath.msg	***string* robot_name**	Name of robot
	***Path* robot_path**	Path of robot
DoorState.msg	***string* name**	Name of door
	***uint8* door_type**	Undefined=0 Single sliding=1 Double sliding=2 Single telescope=3 Double telescope=4 Single swing=5 Double swing=6
	geometry_msgs/Pose	Swinging pose of door
	closed_edge_location	Location of door
	***float32* motion_range**	Range of motion of door
Door.msg	***builtin_interfaces/Time* door_time**	Time at door
	***string* door_name**	Name of door

	***uint8* door_state**	Closed=0
		Moving=1

ROS 2 Message	Details (*message type* message name)	Description
		Opened=2
LiftState.msg	***builtin_interfaces/Time*** **lift_time**	Time at lift
	string **lift_name**	Name of lift
	string[] **available_floors**	Floors available to choose
	string **current_floor**	Current floor the lift is at
	string **destination_floor**	Destination floor the lift is to go

	uint8 **door_state**	Closed =0
		Moving =1
		Opened =2

	uint8 **motion_state**	Stopped =0
		Up =1
		Down =2
		Unknown =3

	uint8[] **available_modes**	Mode of lift

	uint8 **current_mode**	Unknown =0
		Passenger =1
		AGV =2
		Fire =3
		Offline =4
		Emergency =5

	string **session_id**	Session ID
LiftRequest.msg	***string*** **lift_name**	Name of lift
	string **session_id**	Session ID
	uint8 **request_type**	Type of request
	string **destination_floor**	End =0 AGV Mode =1 Passenger Mode =2

	uint8 **door_state**	Opened =2
		Closed =0