OUR EXPERIENCE WITH DEVELOPING RMF FLEET ADAPTER

And some extra tips & tricks with RMF

Managed by Advanced Remanufacturing and Technology Centre ARTC

Overview

- What is RMF?
- Fleet Adapter Template
- Fleet Adapter Development Challenges
- Future plans

What is RMF?

What is RMF?

Fleet Adapter Template

Fleet Adapter Development Challenges

Future plans

What is Robotics Middleware Framework?

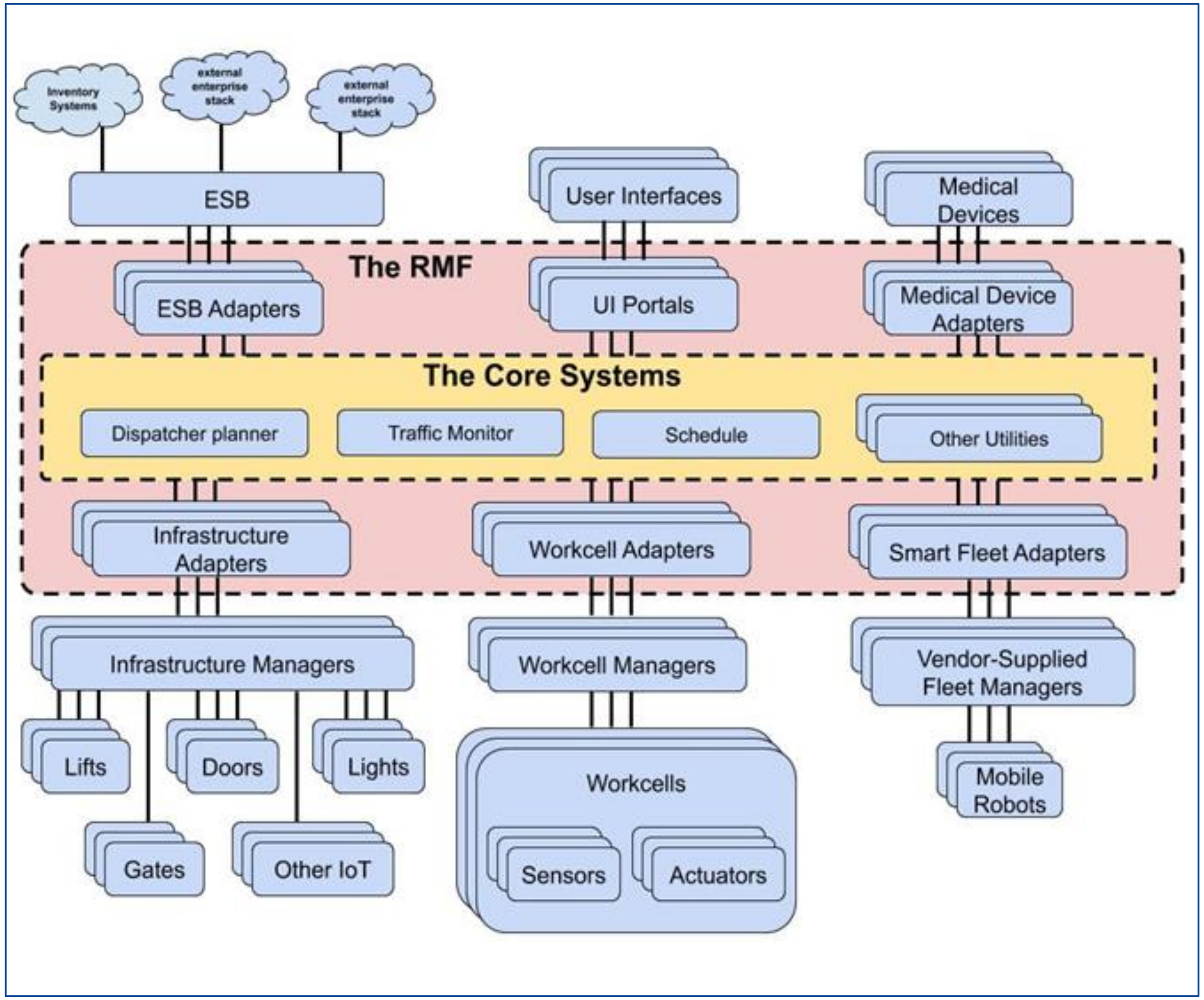
RMF is a collection of software modules and tools that facilitate interoperability among:

- Heterogeneous robot fleets
- Building infrastructure

  (door, lifts, etc.)
- Automated systems

  (dispensers, collectors etc.)

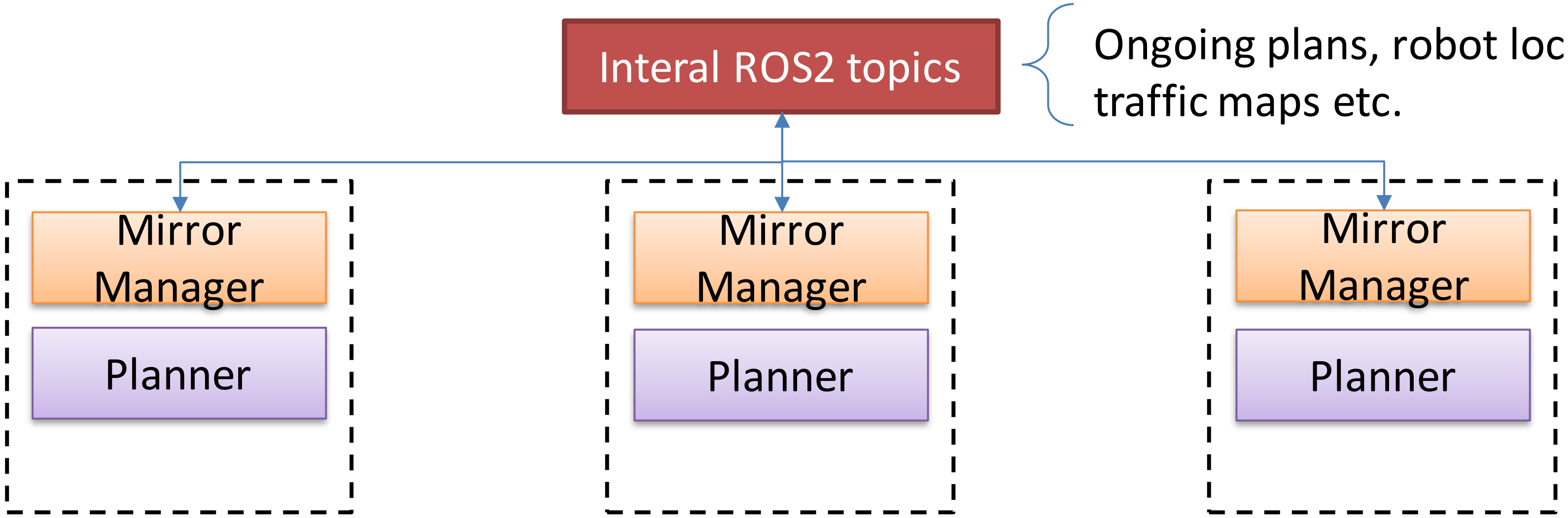
Additionally, it incorporate intelligence into the system such as tasks allocation, resource allocation and traffic management.



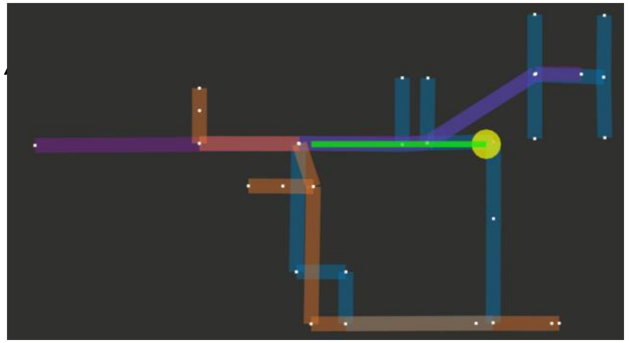
Source: Open Robotics (OSRC)

Traffic Scheduling

- Happens during planning
- Monitor the status of other robots and plan to avoid conflict
  - Delay the plan execution
  - Choose a different path
- Video explanation by Dr Grey (2:30 – 3:12) https://youtu.be/XPZo_dXIXEY

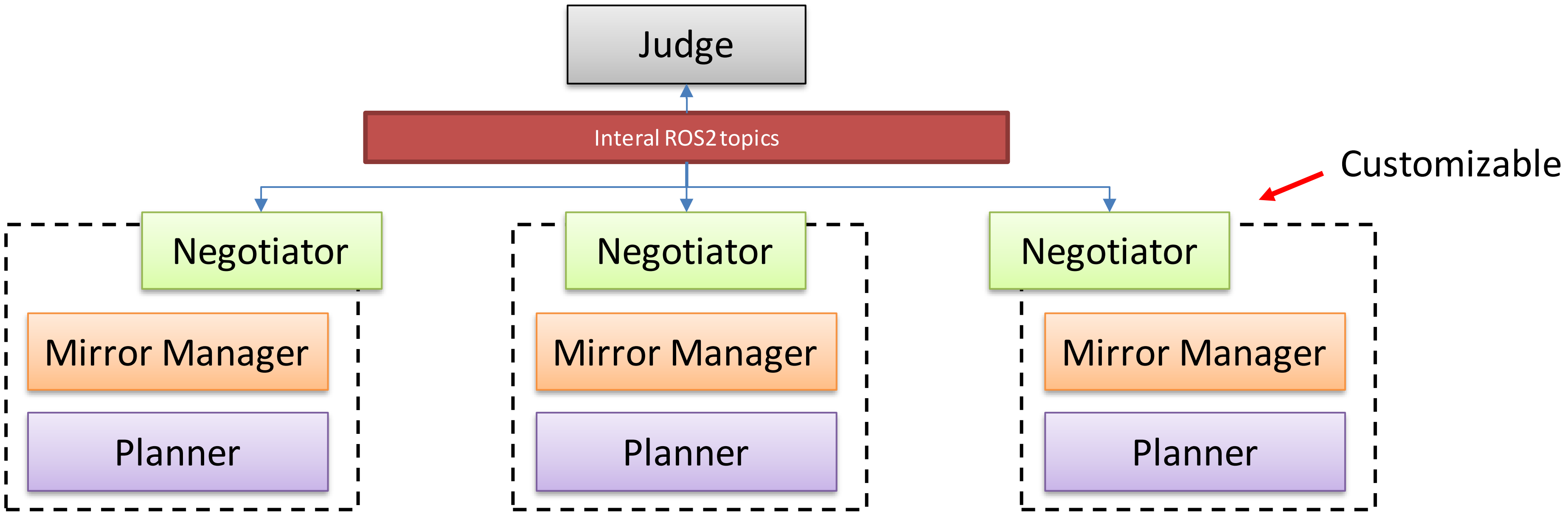


Ongoing plans, robot location, traffic maps etc.

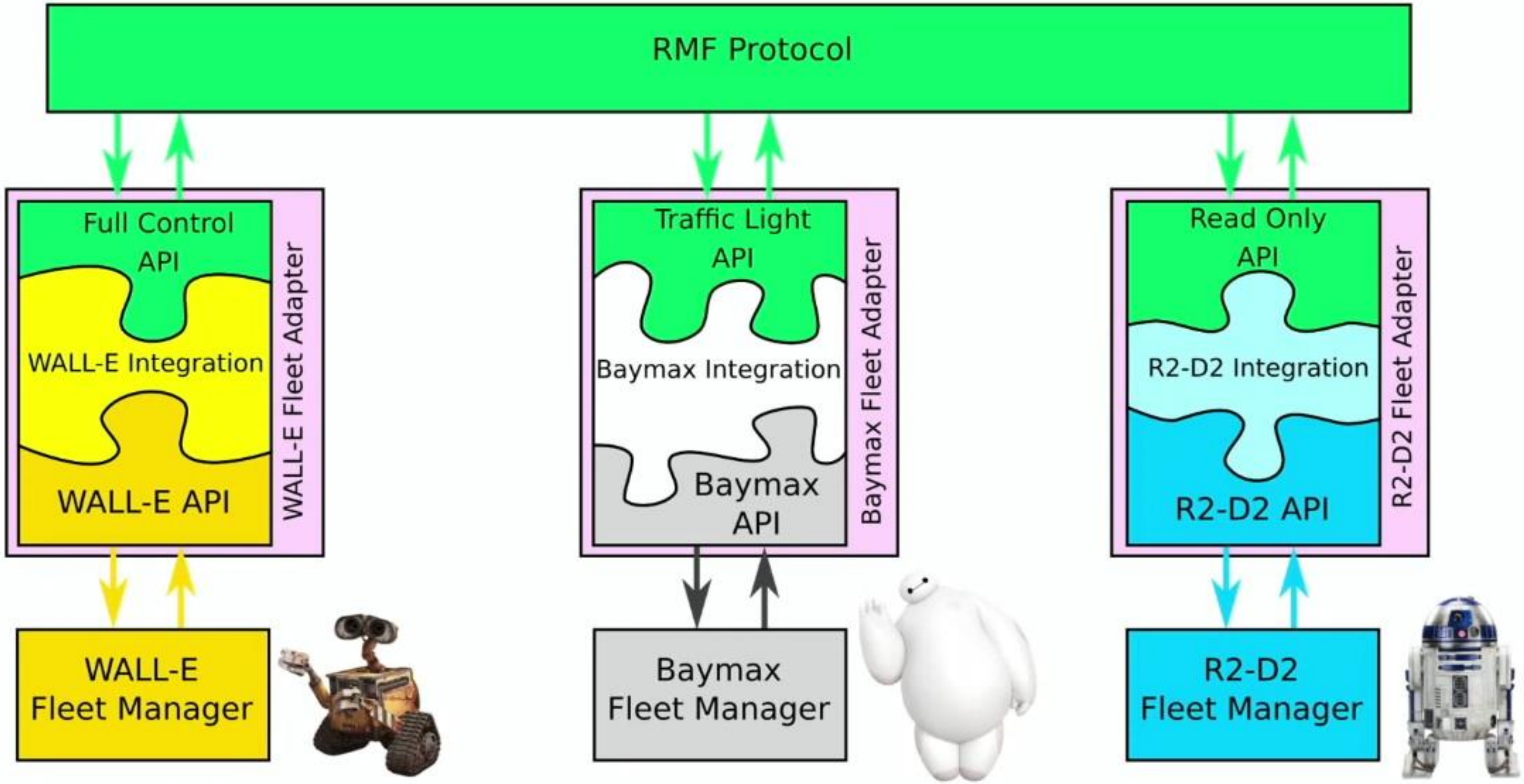


Traffic Negotiation

- Robot does not always go according to plan. (Delay, obstacle avoidance)
- Resolve additional conflict
- Video explanation by Dr Grey (3:56 – 8:40) https://youtu.be/XPZo_dXIXEY



Integration - Adapters



Picture Source: OSRC

Fleet Adapter Template

What is RMF?

Fleet Adapter Template

Fleet Adapter Development Challenges

Future plans

Different Fleet Adapter Design [1/2] – Free Fleet & Simulation

Full control example

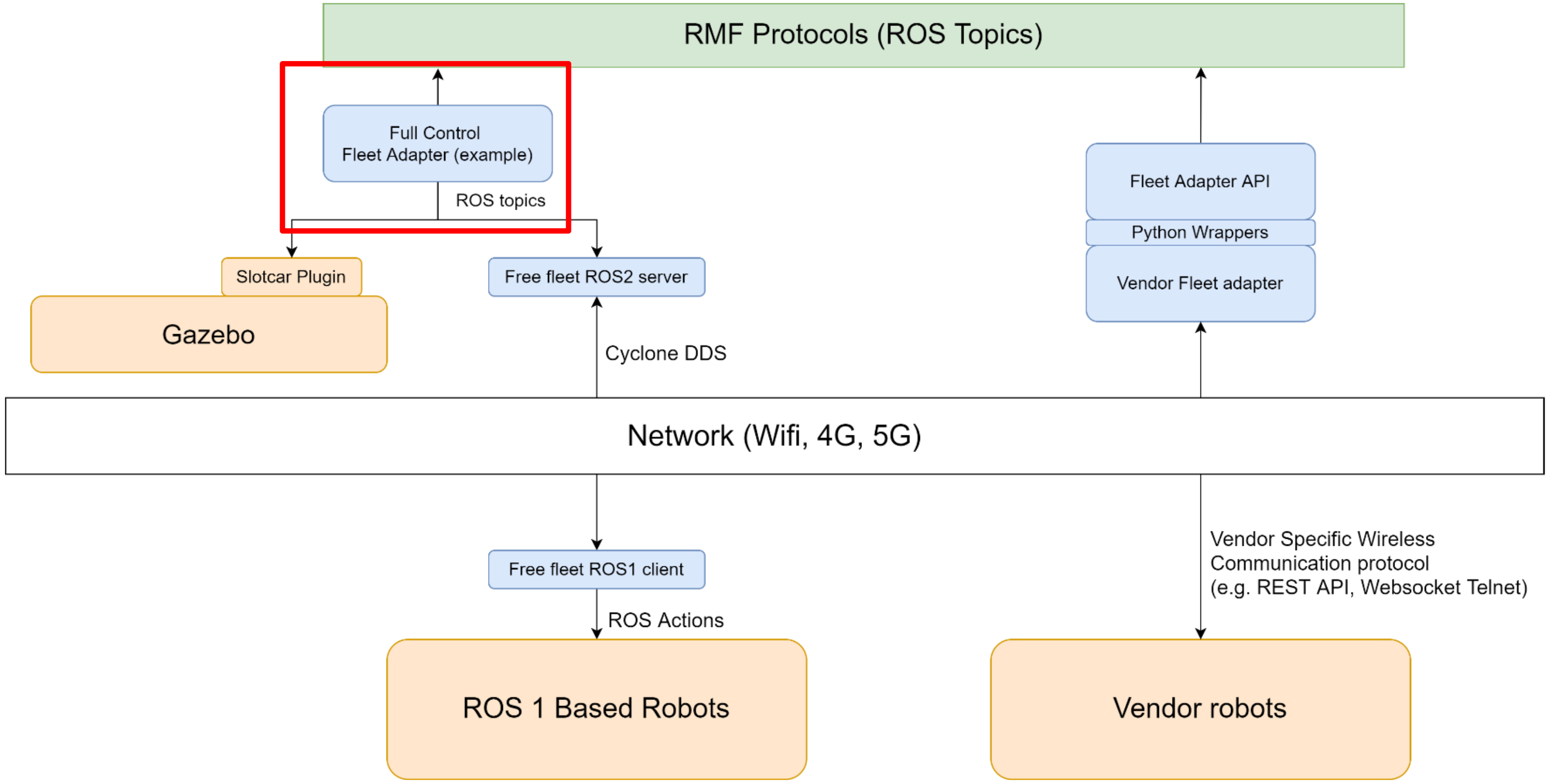
- An example ROS2 node that translate full control fleet adapter API into ROS2 topics.

Slotcar Gazebo Plugin

- A simplified path following plugin to simulate robot motion in Gazebo based on RMF commands.

Free Fleet

- A server and client pair to communicate RMF commands using pure CycloneDDS messages.
- This is plug-and-play for ROS 1 & ROS 2 based robot



Managed by Advanced Remanufacturing and Technology Centre ARTC

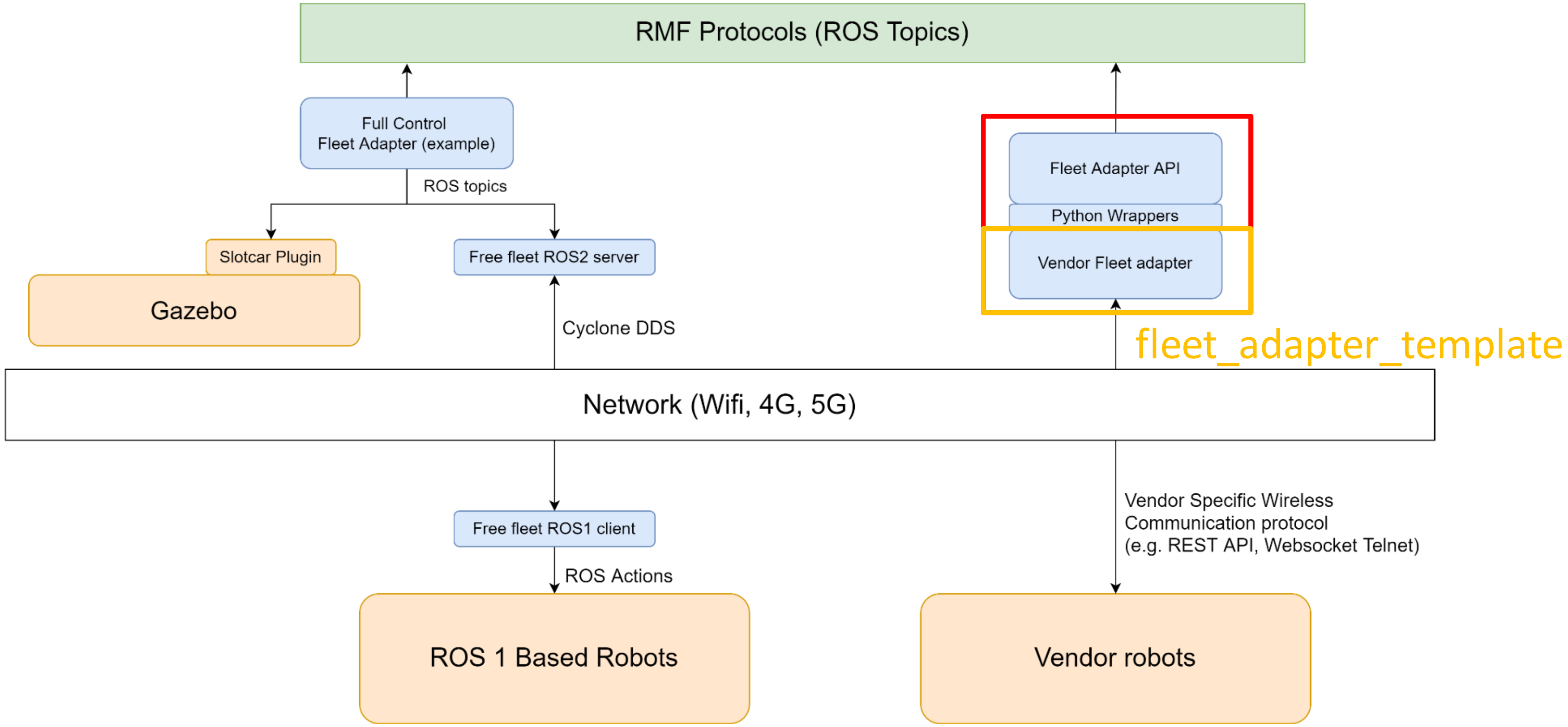
Different Fleet Adapter Design [2/2] – Vendor Fleet Adapter

Python Fleet Adapter Wrapper

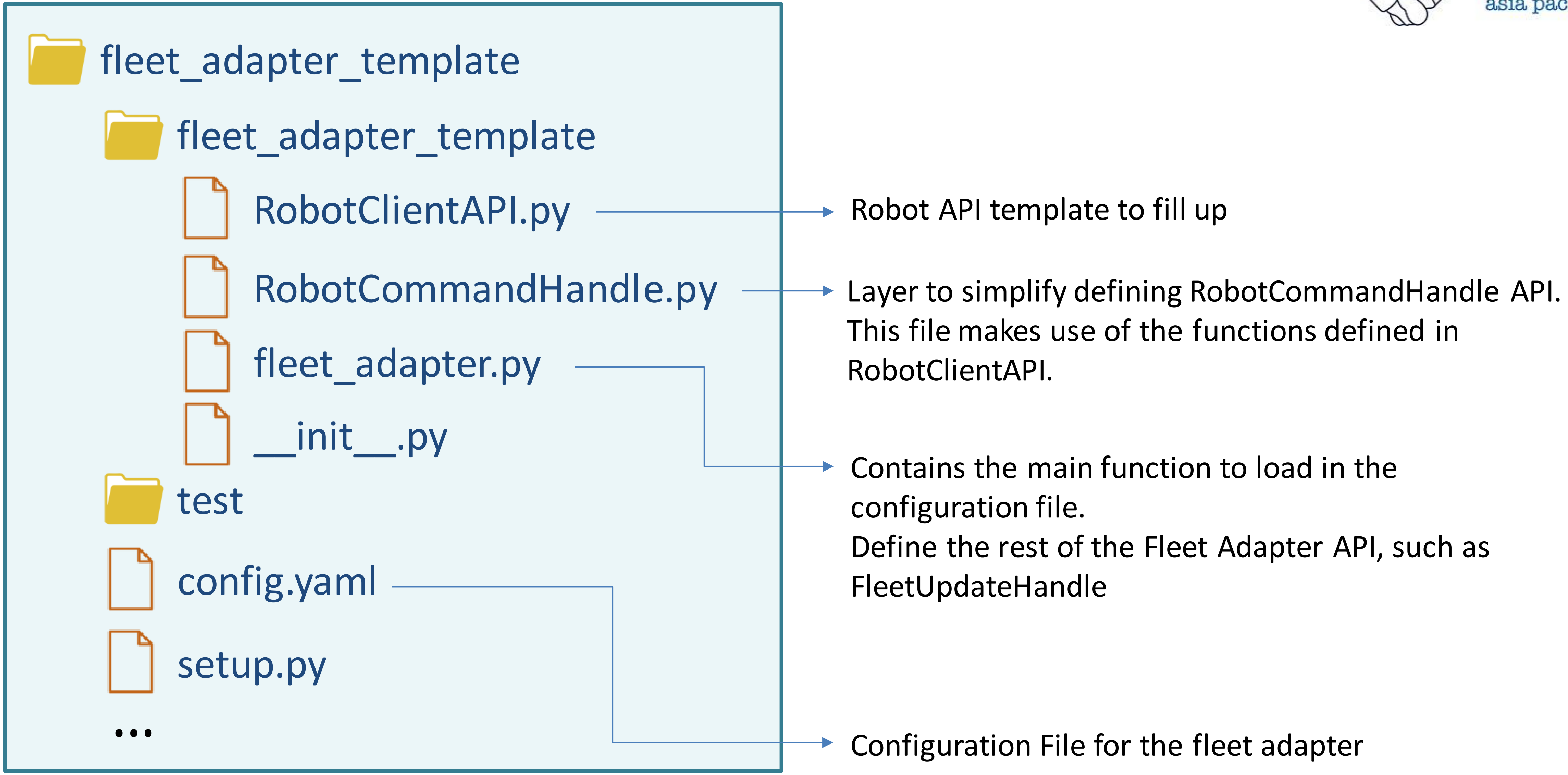
- Python wrapper over the fleet adapter C++ API to offer better ease-of-use

Vendor Fleet Adapter

- Translate fleet adapter API directly into vendor robot specific communication protocol



Fleet Adapter Template – File Structure [1/3]



```
fleet_adapter_template
    fleet_adapter_template
        RobotClientAPI.py
        RobotCommandHandle.py
        fleet_adapter.py
        __init__.py
    test
    config.yaml
    setup.py
    ...
```

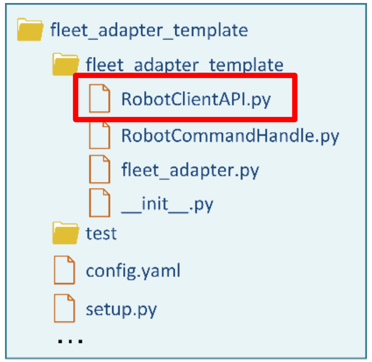
RobotClientAPI.py → Robot API template to fill up

RobotCommandHandle.py → Layer to simplify defining RobotCommandHandle API. This file makes use of the functions defined in RobotClientAPI.

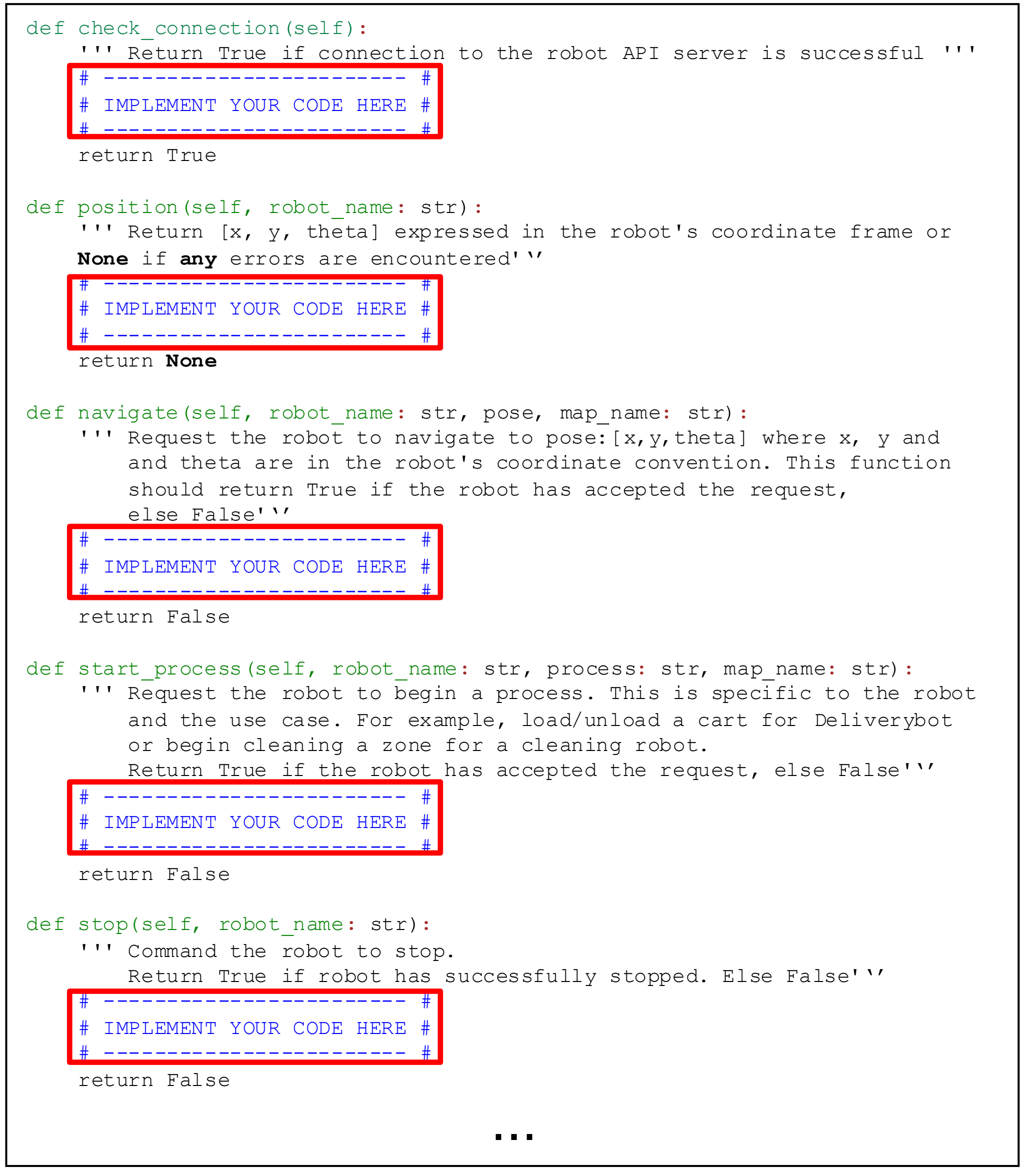
fleet_adapter.py → Contains the main function to load in the configuration file. Define the rest of the Fleet Adapter API, such as FleetUpdateHandle

config.yaml → Configuration File for the fleet adapter

Fleet Adapter Template – RobotClientAPI [2/3]



```
fleet_adapter_template
    fleet_adapter_template
        RobotClientAPI.py
        RobotCommandHandle.py
        fleet_adapter.py
        __init__.py
    test
    config.yaml
    setup.py
...
```



```
def check_connection(self):
    ''' Return True if connection to the robot API server is successful '''
    # ------------------------ #
    # IMPLEMENT YOUR CODE HERE #
    # ------------------------ #
    return True

def position(self, robot_name: str):
    ''' Return [x, y, theta] expressed in the robot's coordinate frame or
    None if any errors are encountered''’
    # ------------------------ #
    # IMPLEMENT YOUR CODE HERE #
    # ------------------------ #
    return None

def navigate(self, robot_name: str, pose, map_name: str):
    ''' Request the robot to navigate to pose:[x,y,theta] where x, y and
        and theta are in the robot's coordinate convention. This function
        should return True if the robot has accepted the request,
        else False''’
    # ------------------------ #
    # IMPLEMENT YOUR CODE HERE #
    # ------------------------ #
    return False

def start_process(self, robot_name: str, process: str, map_name: str):
    ''' Request the robot to begin a process. This is specific to the robot
        and the use case. For example, load/unload a cart for Deliverybot
        or begin cleaning a zone for a cleaning robot.
        Return True if the robot has accepted the request, else False''’
    # ------------------------ #
    # IMPLEMENT YOUR CODE HERE #
    # ------------------------ #
    return False

def stop(self, robot_name: str):
    ''' Command the robot to stop.
        Return True if robot has successfully stopped. Else False''’
    # ------------------------ #
    # IMPLEMENT YOUR CODE HERE #
    # ------------------------ #
    return False

                        ...
```

Fill in the code using

vendor specific client library

Fleet Adapter Template – Configuration [3/3]

```
robots:
  # Here the user is expected to append the configuration for each robot in the
  # fleet.
  # Configuration for first robot in this fleet
  deliverybot1:
    robot_config:
      max_delay: 10.0 # allowed seconds of delay of the current itinerary before it gets interrupted and replanned
    rmf_config:
      robot_state_update_frequency: 0.5
      start:
        map_name: "L1"
        # waypoint: "charger_deliverybot1" # Optional
        # orientation: 0.0 # Optional, radians
        waypoint: null
        orientation: null
      charger:
        waypoint: "charger_deliverybot1"
```

Robot specific settings

```
reference_coordinates:
  rmf:
    - [20.33, -3.156]
    - [8.908, -2.57]
    - [13.02, -3.601]
    - [21.93, -4.124]
  robot:
    - [59, 399]
    - [57, 172]
    - [68, 251]
    - [75, 429]
```

Coordinate transform between

RMF navigation graph

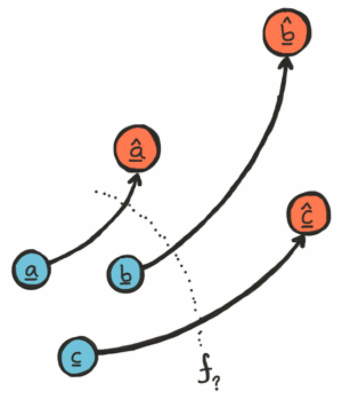
and

the robot internal SLAM map

Managed by Advanced Remanufacturing and Technology Centre ARTC

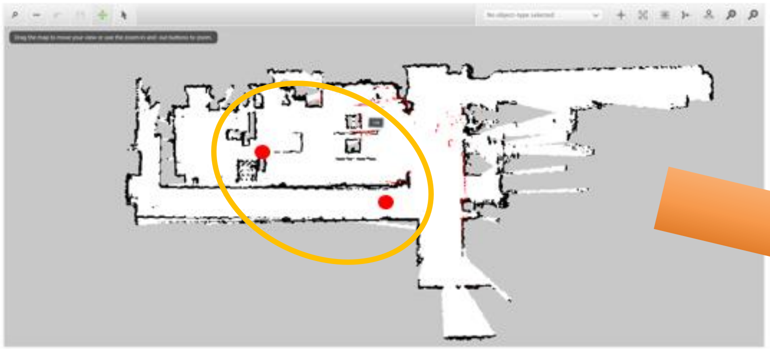
Coordinate transform – How it works

Python Nudged Library

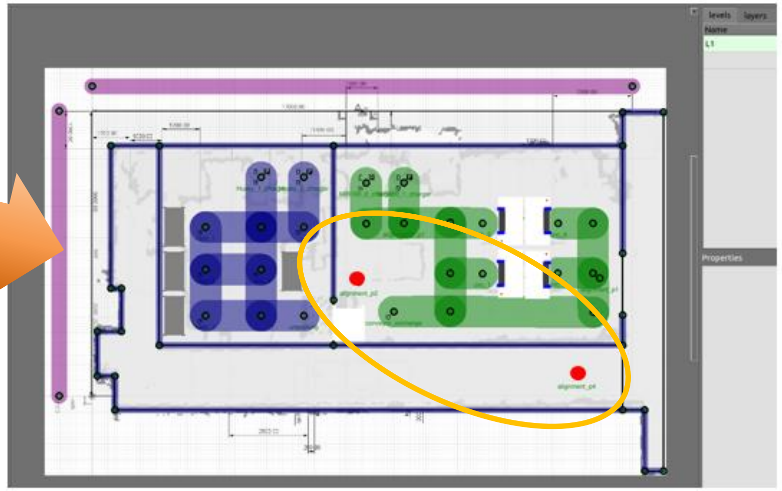


```
reference_coordinates:
  rmf:
    - [20.33, -3.156]
    - [8.908, -2.57]
    - [13.02, -3.601]
    - [21.93, -4.124]
  robot:
    - [59, 399]
    - [57, 172]
    - [68, 251]
    - [75, 429]
```

Alignment point in MiR Map



Alignment Point in RMF Traffic Map



Ready the coordinates from traffic editor

Fleet Adapter Development Challenges

Omron Fleet Adapter Development Lessons Learnt

What is RMF?

Fleet Adapter Template

Fleet Adapter Development Challenges

Future plans

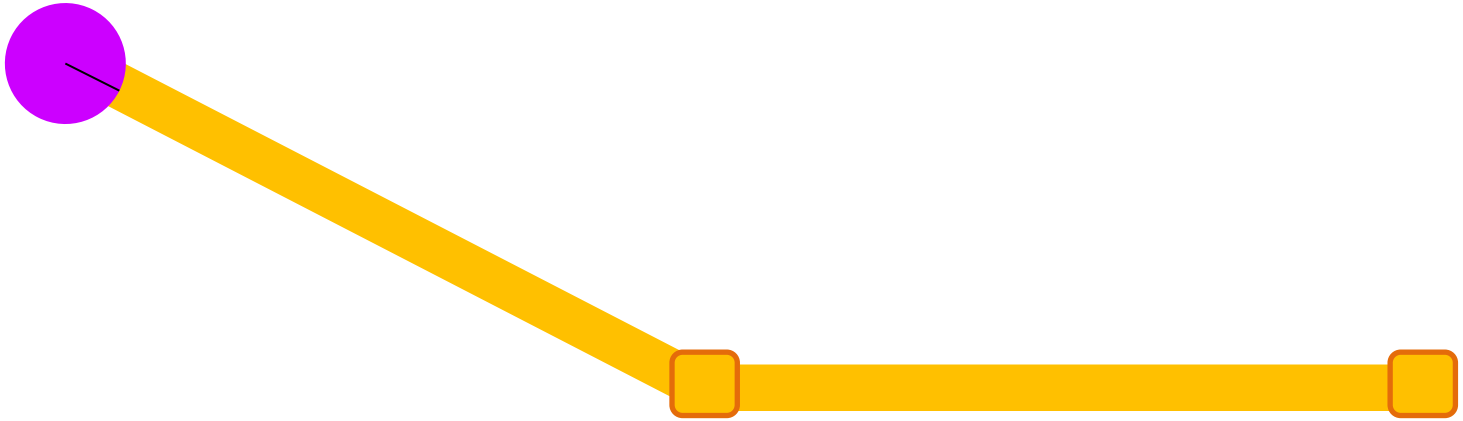
Challenges - Summary

- Robot Orientation
- Redundant robot behavior
- Multi-level navigation
- Customized interaction with workcell
  - Dispenser & Ingestor
  - Docking function

Challenges – robot orientation [1/2]

Issue 1: Robot keeps turning to the wrong / same direction after reaching a waypoint

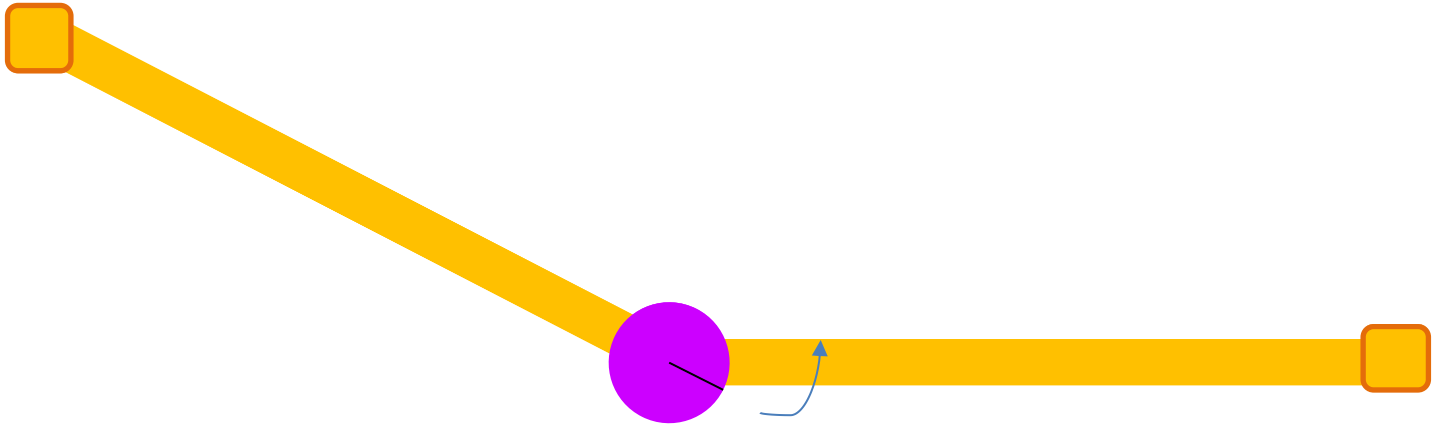
Check against degree to radian conversion in the fleet adapter.
If the robot only accepts Degrees (0 - 360) while the command sent to it is Radians(0 – 3/3.14), the robot might interpret as always heading to the same location.



Challenges – robot orientation [1/2]

Issue 1: Robot keeps turning to the wrong / same direction after reaching a waypoint

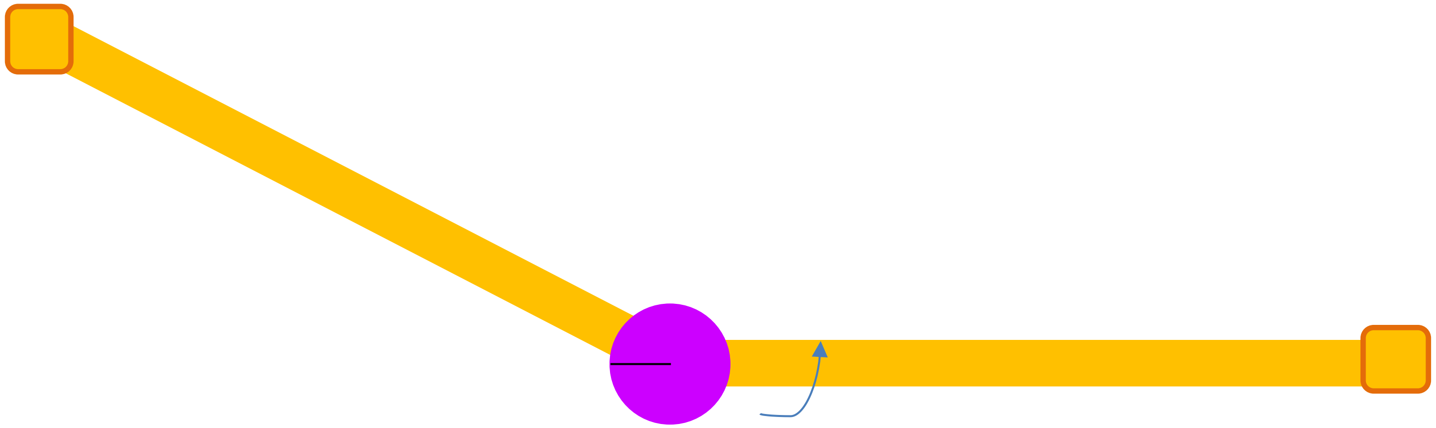
Check against degree to radian conversion in the fleet adapter.
If the robot only accepts Degrees (0 - 360) while the command sent to it is Radians(0 – 3/3.14), the robot might interpret as always heading to the same location.



Challenges – robot orientation [1/2]

Issue 1: Robot keeps turning to the wrong / same direction after reaching a waypoint

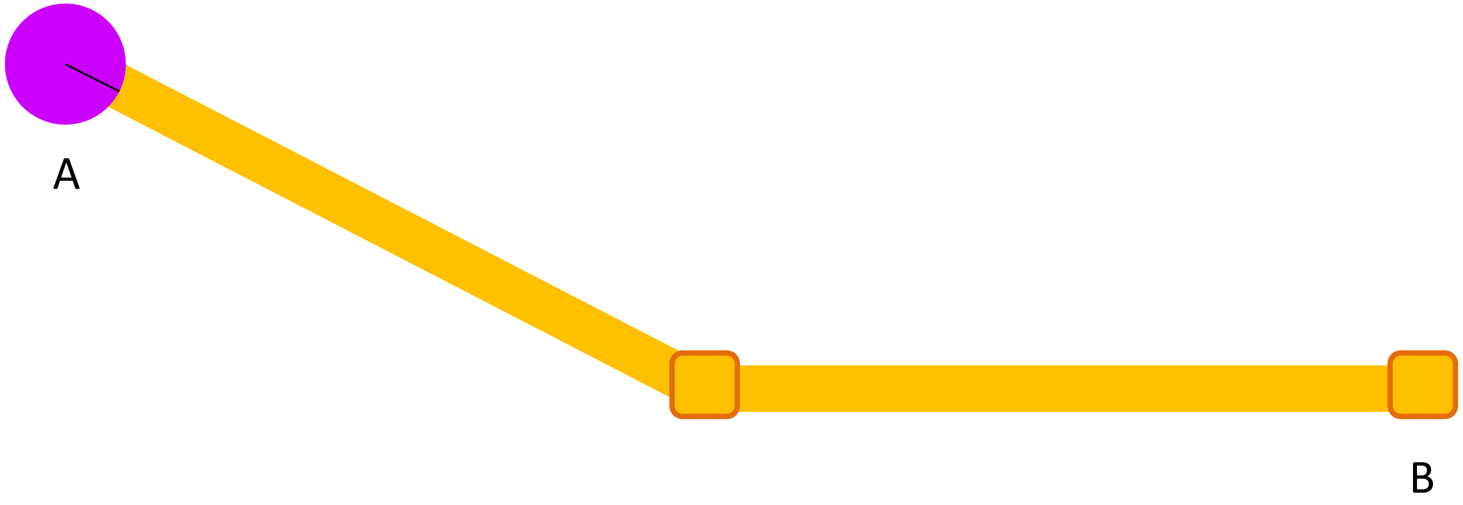
Check against degree to radian conversion in the fleet adapter.
If the robot only accepts Degrees (0 - 360) while the command sent to it is Radians(0 – 3/3.14), the robot might interpret as always heading to the same location.



Challenges – robot orientation [2/2]

Issue 2: Robot end orientation at each point

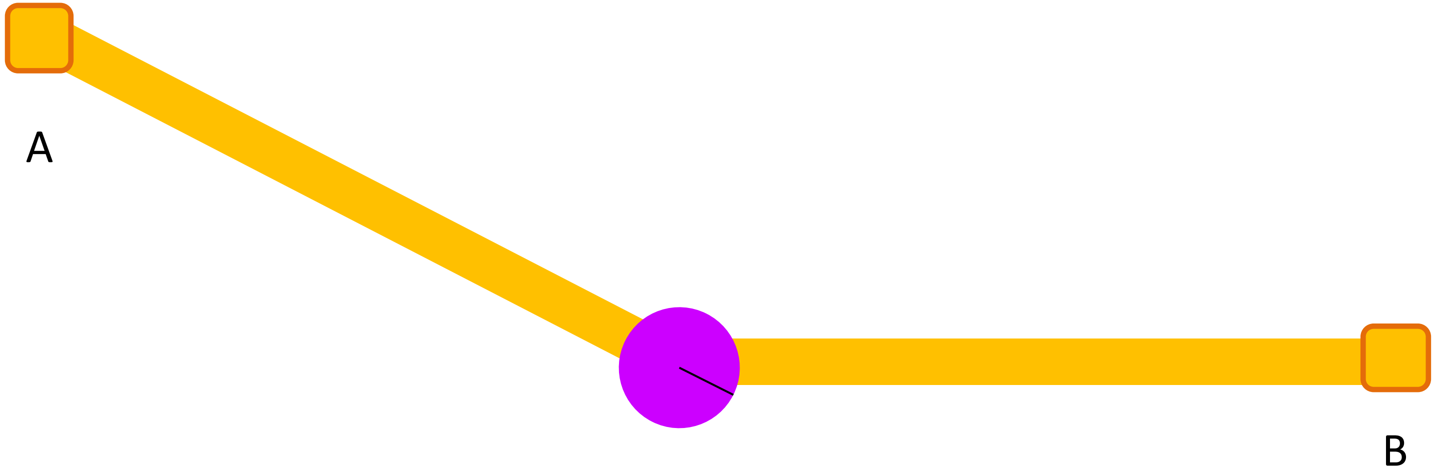
If using Loop Task last from A to B, robot heading will be pointing towards the last direction it should be traveling in



Challenges – robot orientation [2/2]

Issue 2: Robot end orientation at each point

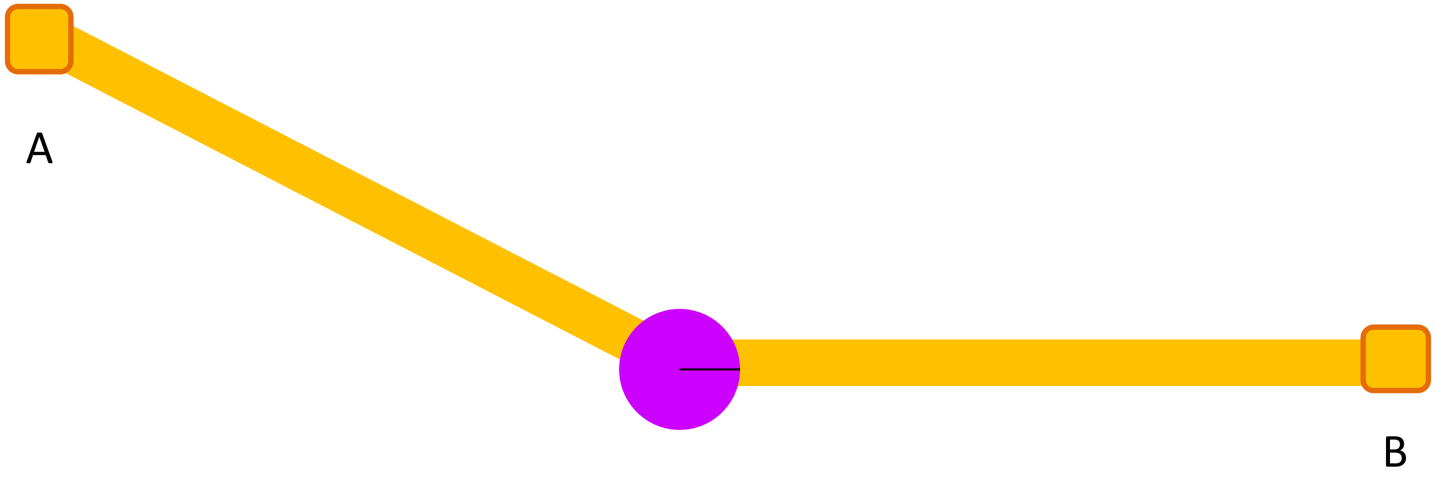
If using Loop Task last from A to B, robot heading will be pointing towards the last direction it should be traveling in



Challenges – robot orientation [2/2]

Issue 2: Robot end orientation at each point

If using Loop Task last from A to B, robot heading will be pointing towards the last direction it should be traveling in



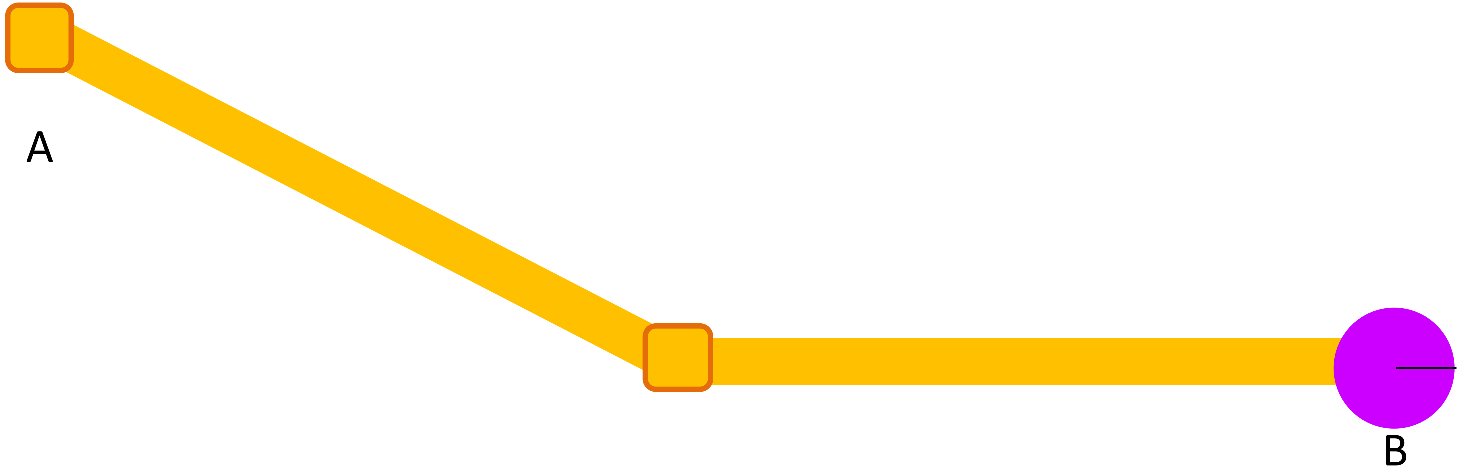
Challenges – robot orientation [2/2]

Issue 2: Robot end orientation at each point

If using Loop Task last from A to B, robot heading will be pointing towards the last direction it should be traveling in

Solution / Workaround:

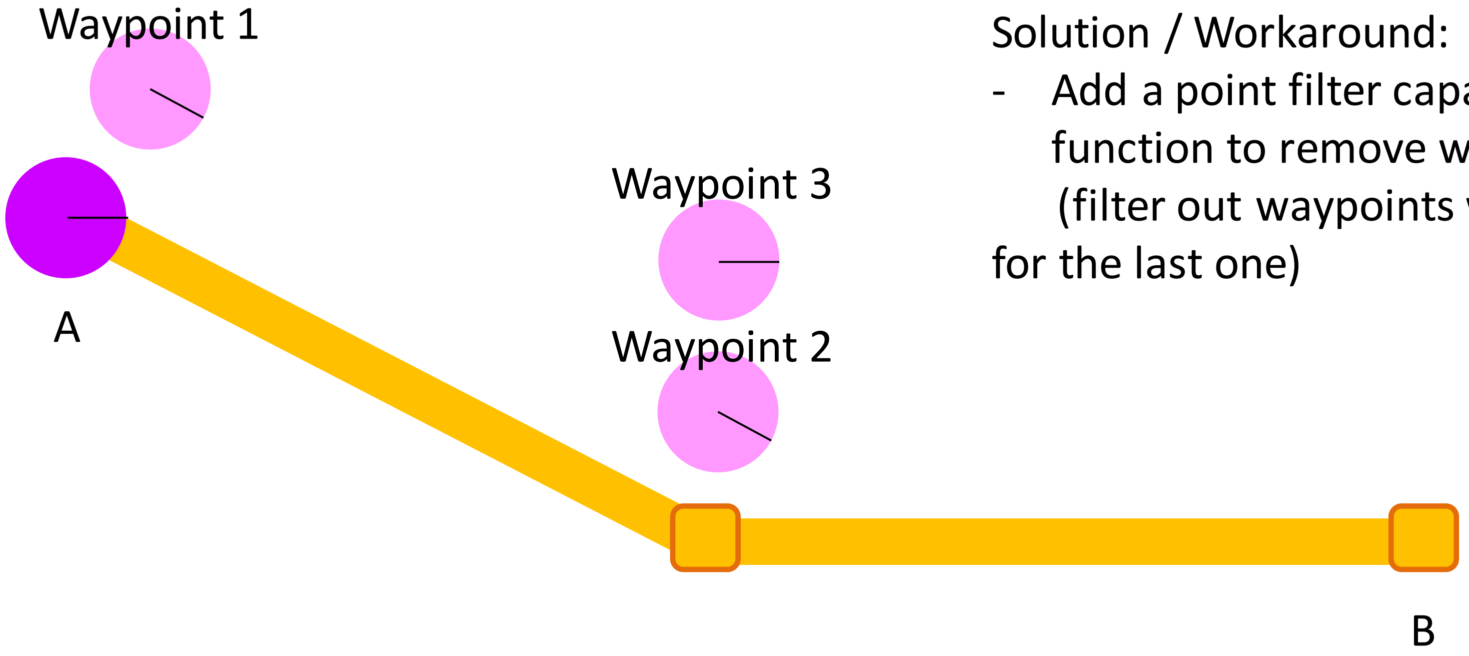
- Change B into a dock and defining a docking action
- Send a GoToPlace task instead of a Loop Task (Re-defining the task sequence)



Challenges – Redundant Motion

Issue: Redundant Robot Motion for point to point navigation

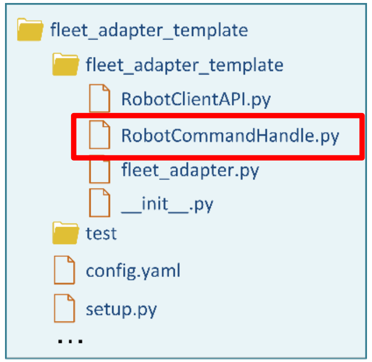
Extra waypoints are sent to the robot for orientation correction.



Only Waypoint 3 is needed

Solution / Workaround:

- Add a point filter capability in follow_new_path() function to remove waypoint 1 & 2.
  (filter out waypoints with similar graph_index except for the last one)



```
fleet_adapter_template
    fleet_adapter_template
        RobotClientAPI.py
        RobotCommandHandle.py
        fleet_adapter.py
        __init__.py
    test
    config.yaml
    setup.py
    …
```

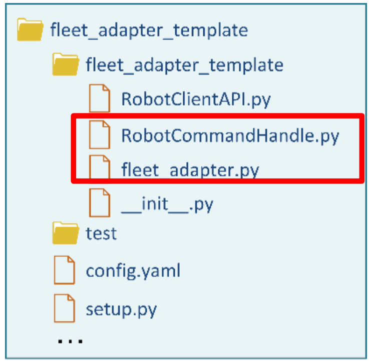
Challenges – Multi-Level Navigation

Issue: Not supported by default in the template

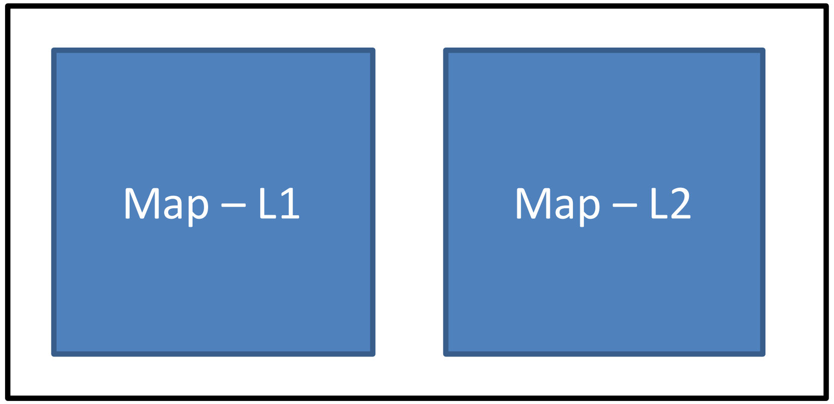
Changes are needed in RobotCommandHandle.py and fleet_adapter.py

Below describe the behavior for Omron fleet adapter

1. Waypoint with a different level is detected
2. Localize the robot to a different location
3. Start navigate to the waypoint (using a new set of coordinate transform)



```
fleet_adapter_template
    fleet_adapter_template
        RobotClientAPI.py
        RobotCommandHandle.py
        fleet_adapter.py
        __init__.py
    test
    config.yaml
    setup.py
    …
```



SLAM Map

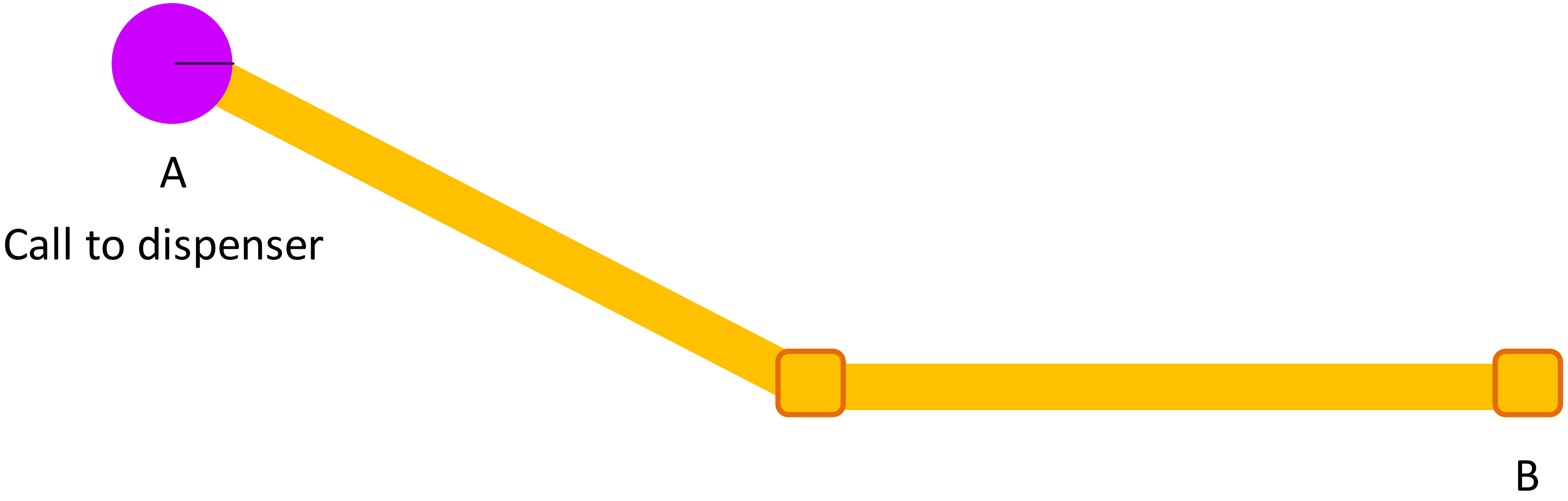
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 1:

Delivery Task

1. Define the required dispenser and ingestor pair.
2. Use a dummy dispenser / ingestor if no action is needed.



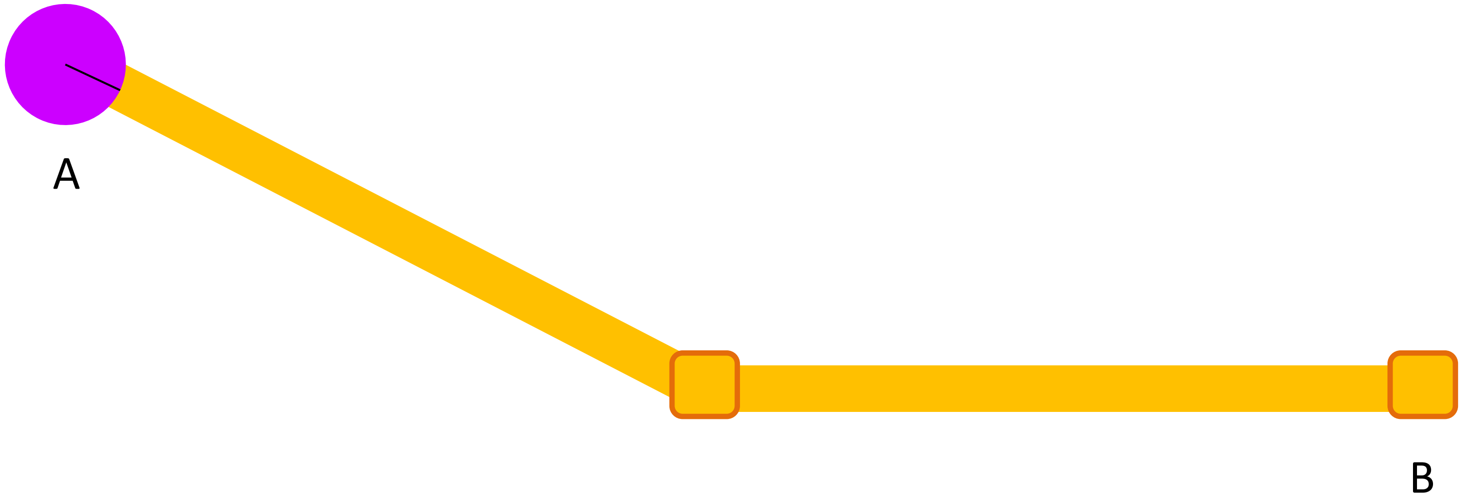
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 1:

Delivery Task

1. Define the required dispenser and ingestor pair.
2. Use a dummy dispenser / ingestor if no action is needed.



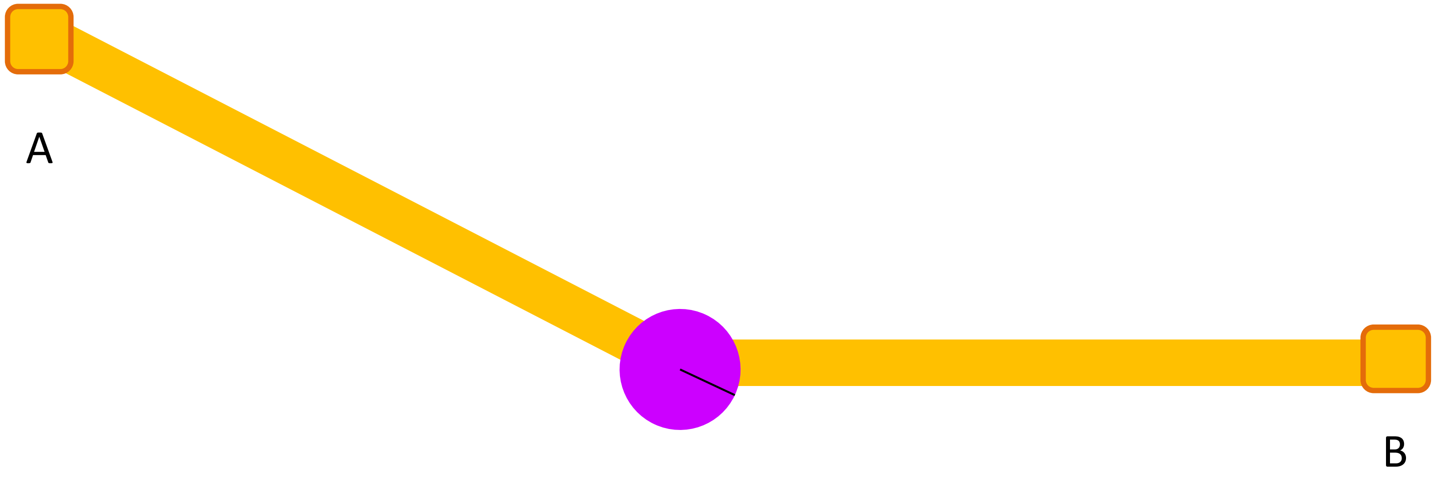
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 1:

Delivery Task

1. Define the required dispenser and ingestor pair.
2. Use a dummy dispenser / ingestor if no action is needed.



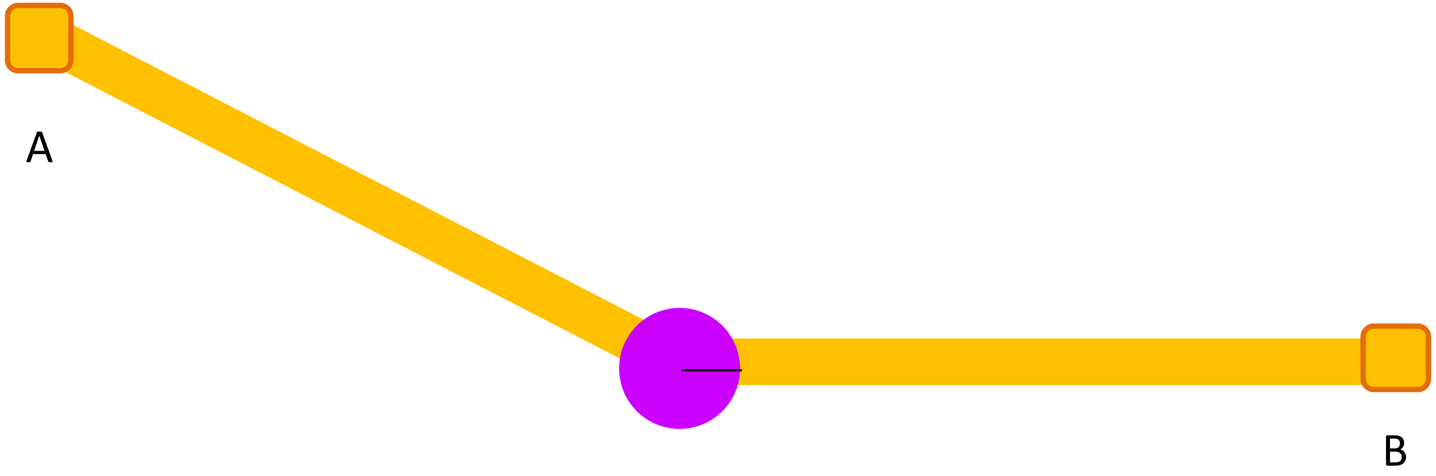
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 1:

Delivery Task

1. Define the required dispenser and ingestor pair.
2. Use a dummy dispenser / ingestor if no action is needed.



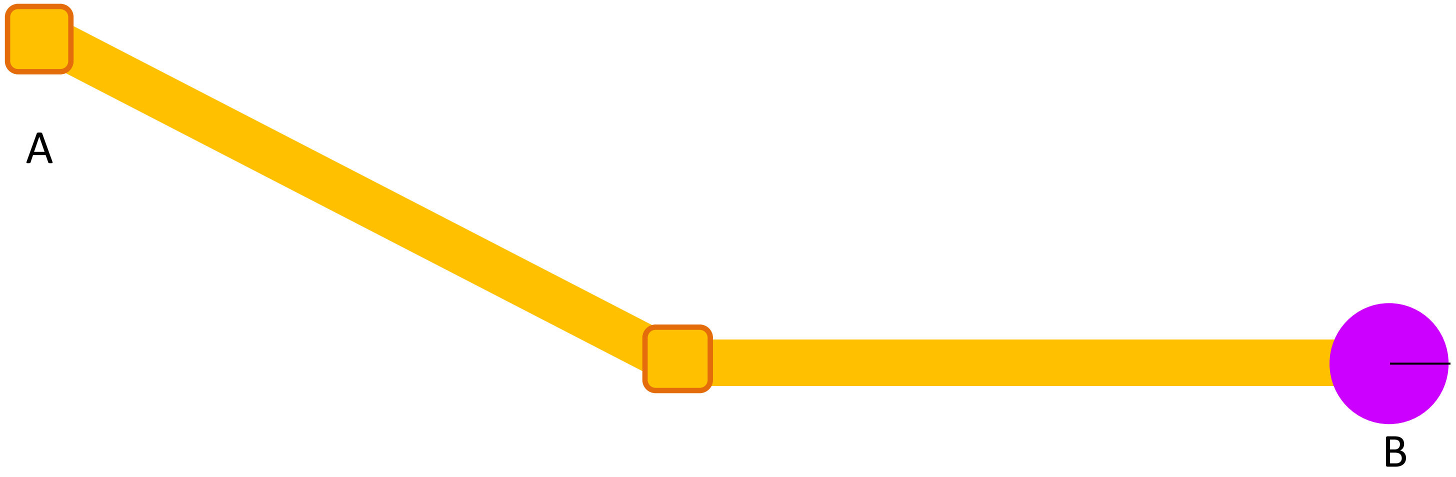
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 1:

Delivery Task

1. Define the required dispenser and ingestor pair.
2. Use a dummy dispenser / ingestor if no action is needed.



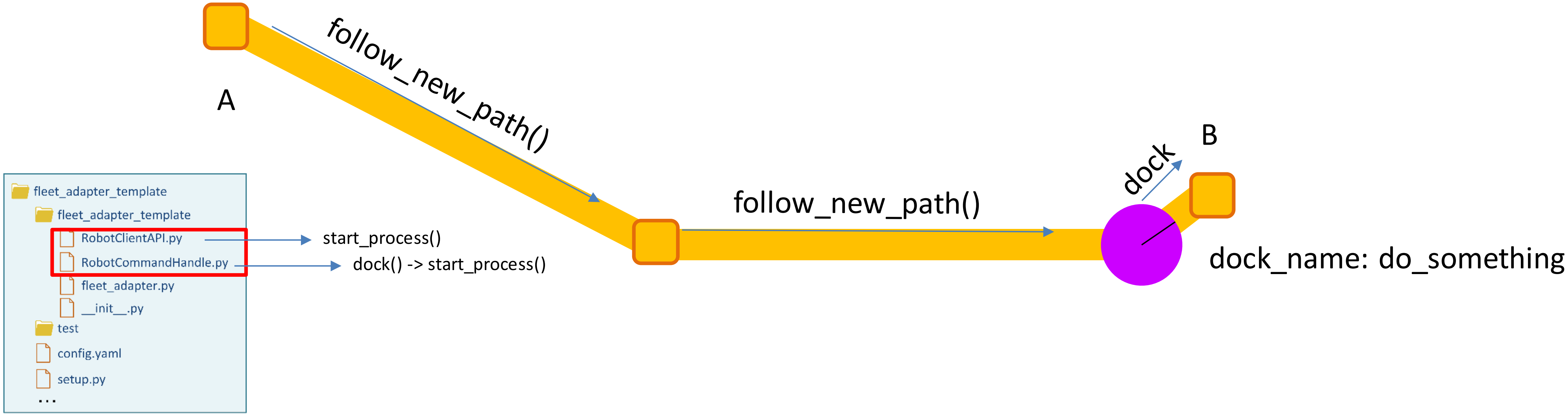
Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Solution 2:

Define a dock

1. Change the RMF traffic map
2. Define the start_process() function to handle “dock_name” in RobotClientAPI.py
3. Update the dock() function if needed.



Challenges – Customized Interaction with Workcell

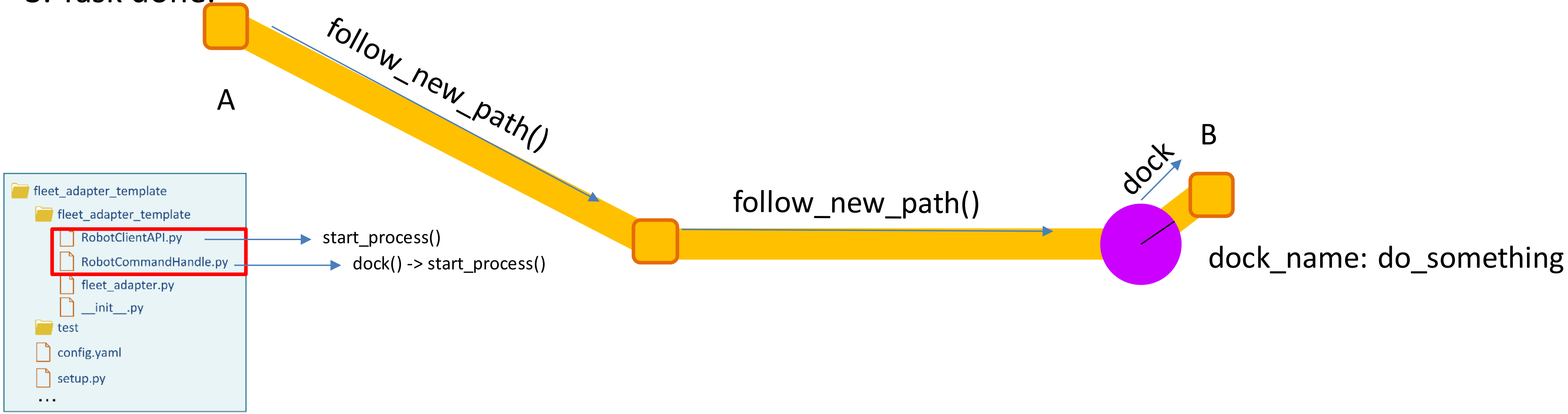
Issue: Robot cannot call customized command to interaction with the workcell

Solution 2:

Define a dock

Behavior:

1. Robot navigate to the waypoint before the "dock" – waypoint with a dock_name defined.
2. Calls the start_process(do_something) function
3. Task done.



```
fleet_adapter_template
    fleet_adapter_template
        RobotClientAPI.py
        RobotCommandHandle.py
        fleet_adapter.py
        __init__.py
    test
    config.yaml
    setup.py
    …
```

RobotClientAPI.py → start_process()
RobotCommandHandle.py → dock() -> start_process()
A
follow_new_path()
follow_new_path()
dock
B
dock_name: do_something

Challenges – Customized Interaction with Workcell

Issue: Robot cannot call customized command to interaction with the workcell

Summary:

Using dispenser & ingestor

- No additional changes to fleet adapter
- Cannot control robot orientation
- Workaround such as using a dummy dispenser / ingestor might be needed

Using dock

- Changes to fleet adapter
- Can control robot orientation
- Dock function might be subject to change in the near future

Fleet Adapter Development Challenges

Omron Fleet Adapter Development Lessons Learnt

What is RMF?

Fleet Adapter Template

Fleet Adapter Development Challenges

Future plans

What to expect in the next month

Open Source Release

- Fleet Adapter Omron beta release (date: TBD)
- Kone Lift Cloud API adapter beta release (date: TBD)

More in the future

- EasyFullControl API (OSRC)
- Standardization of fleet adapter capability evaluation

Q&A

Thank You!