Temporal heat map of human occupancy

Better Factory

Part A

Objective of temporal heat map

- Compute probability of a floor being occupied by human for a given hour
- This can be used to
    - Plan AGV routes considering that probability from the get go
    - Determine where to put additional Sensors that track humans in a safe way

How does it work at conceptual Level?

Laser Scanner scans floor every second with every laser scanner overlapping its scans as illustrated in the Figure 1.

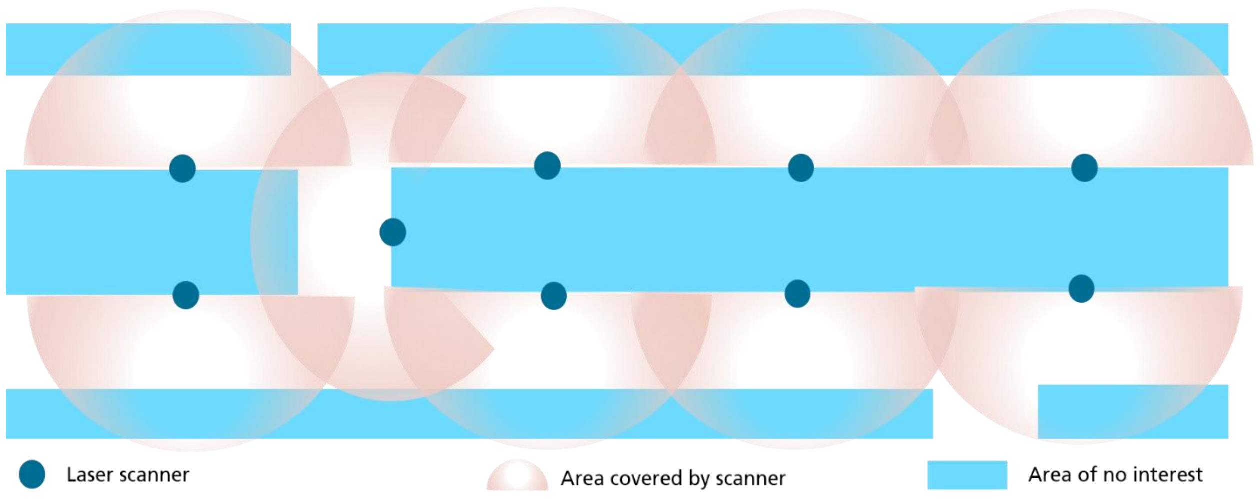


Figure 1: Illustration of the physical demonstrator

Occupancy grid of the floor is computed by unifying the scan data from the laser scanners. Occupancy grid is then aggregated to 1 hour intervals to get the probability of human occupancy in our region of interest. Figure 2 illustrates the 1 hour interval aggregated occupancy grid where the blue circles represents static objects that has occupied the cell for 100% of the time and the red circle represents the human occupancy in the cells at the T junction. The occupancy of cells at the T junction varies from 10% to 60% over 1 hour interval.

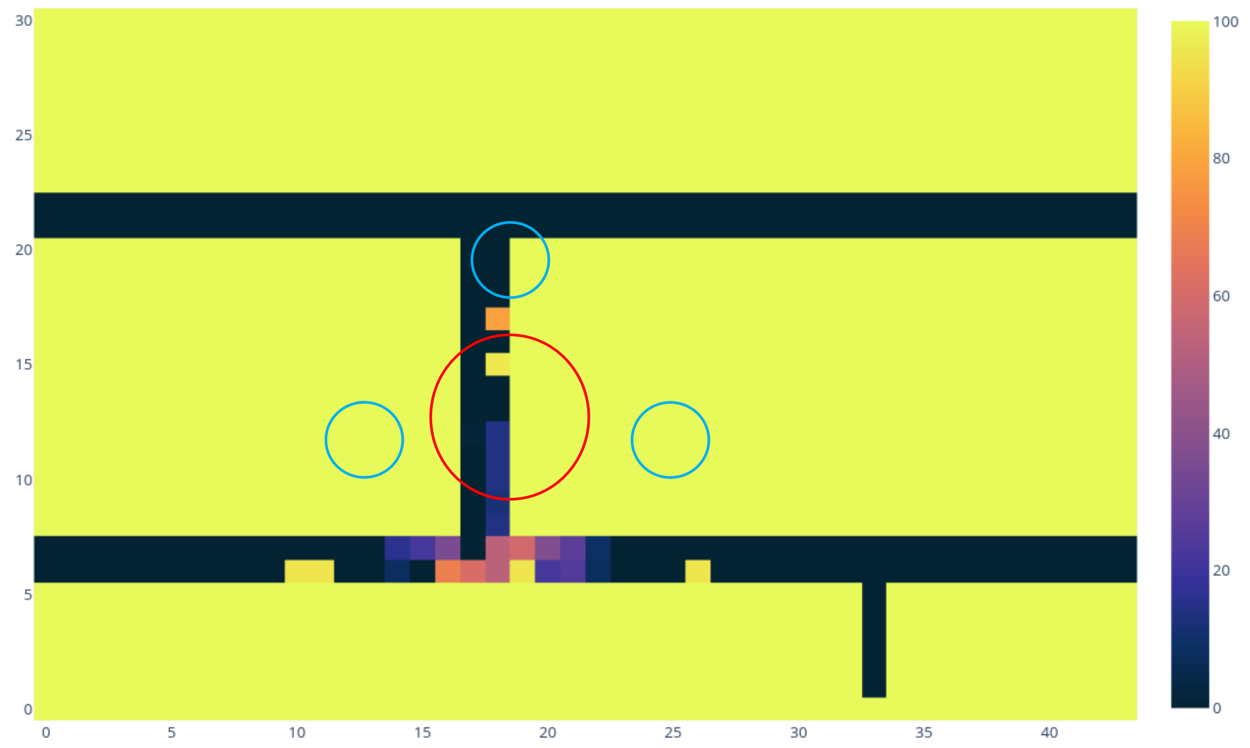


Figure 2: Occupancy grid aggregated to 1 hour intervals

Figure 3 illustrated the high level system architecture of THMHO system. The components responsible for scanning the floors and aggregating the occupancy grid is implemented using the ROS framework. The aggregated heat maps are published to the Orion context broker at aggregation intervals to make it available for the RAMP apps to consume the data. The occupancy grid is stored in DB on premises, where other modules can query the information through an API endpoint. The data stored in the database can be also visualized using the custom made dashboard for visual inspection of the heat maps.

Hardware Requirements

List of hardware components

Component name	**quantity**
Raspberry Pi 4	9
Slamtec RP Lidar A2	9
Intel NUC	1

Table 1: List of components

We have used 9 laser scanners for testing our solution but only 3 of them have 3D printed casings. The listed hardware can be provided by IPA for the evaluation purposed for a finite period of time. We can provide the additional 6 scanners for the evaluation, once 3D printing of casing is completed

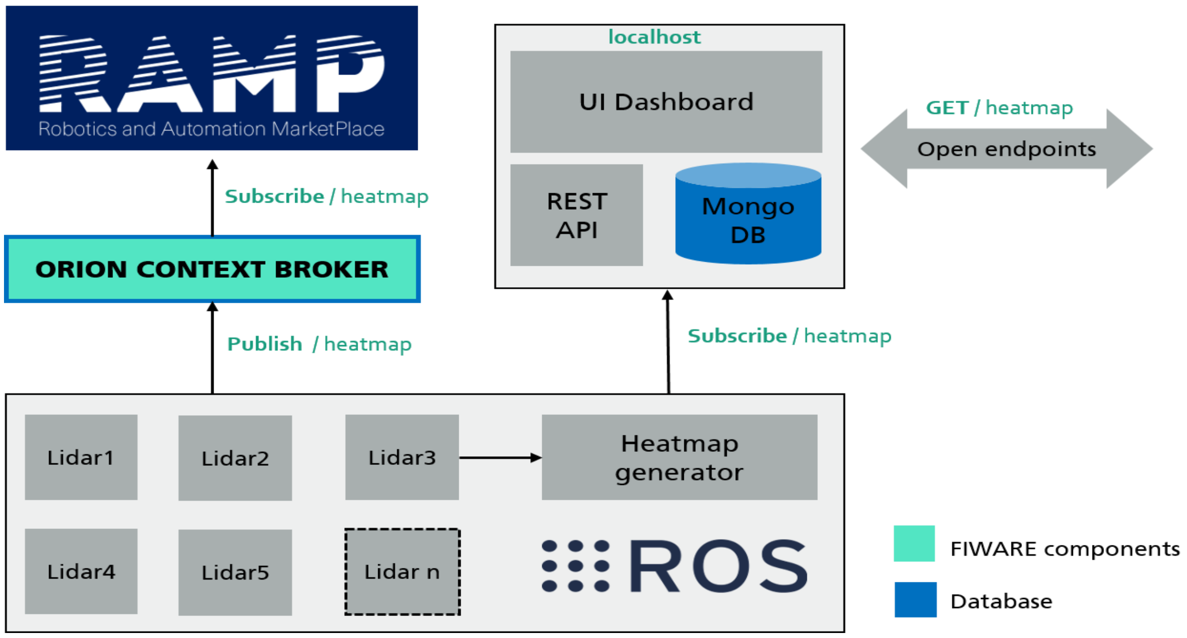


Figure 3: High level overview of the THMHO system architecture

Part B

Software repository?

1. RP Lidar node - THMHO_lidar_node
2. Unify laser scans to generate the heat map - THMHO_heatmap_generator
3. Occupany grid message to NGSIv2 standard message conversion for orion context broker - THMHO_ngsi_converter_bringup
4. REST API and dashboard to get the heat maps
5. Deployment of software components in the distributed system auto-deploy-edge-nodes

Deployment of the components?

The THMHO system is designed as a distributed system and the deployment process is made easier using Ansible.

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. Technically, Ansible is an agentless automation tool that you install on a control node. From the control node, Ansible can manage machines and other devices remotely over SSH protocol.

Directory layout for the deployment repository is shown in the table below. The tasks to be executed are provided in *yml* file,s which will be run on the target locations provided by *hosts* file. The deployment is first tested on the staging environment and then tested for deployment issues and then deployed in the production environment. The components are deployed in Docker containers and therefore it is easier to scale up the number of nodes for scanning and also rolling out updates.

```
inventories/
   production/
      hosts                   # inventory file for production servers
      node1/
        run.sh
      node .. n/              # configuration file for production nodes
        run.sh
   staging/
      hosts                   # inventory file for staging environment

roles/
    tasks/
       install_docker.yml # install docker in production nodes
       copy_exec_file.yml # copy the configuration files for nodes
       pull_run_docker_container.yml # run docker container in the nodes

ansible.cfg
```

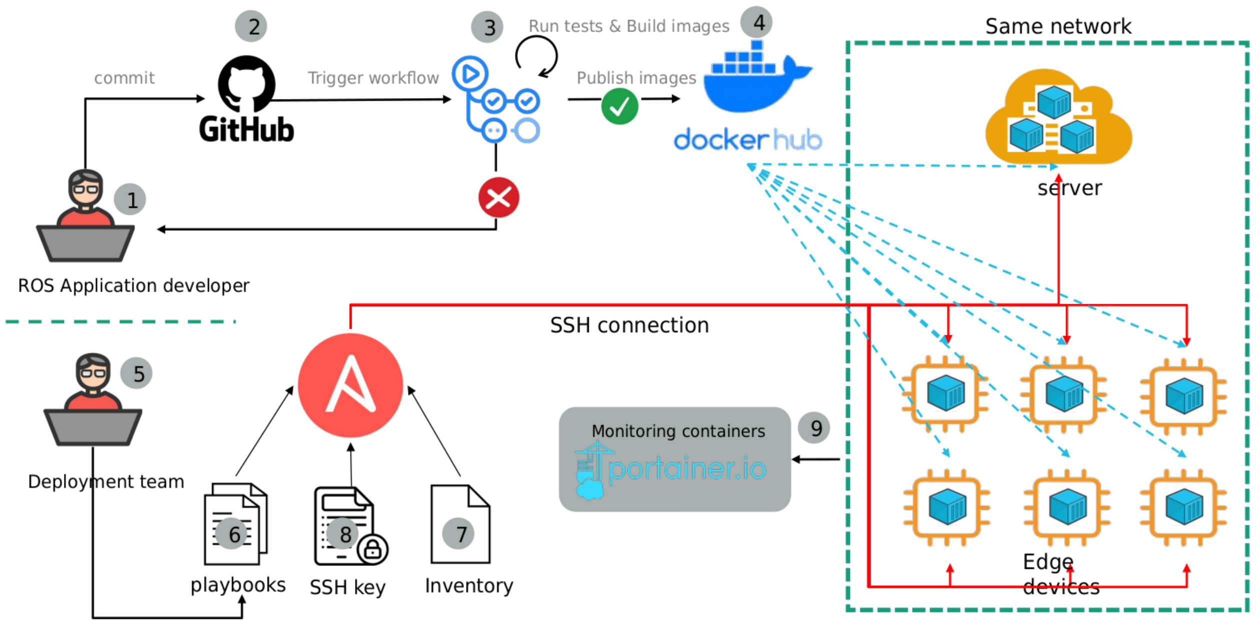


Figure 4: DevOps pipeline overview for the deployment in edge and cloud on premises

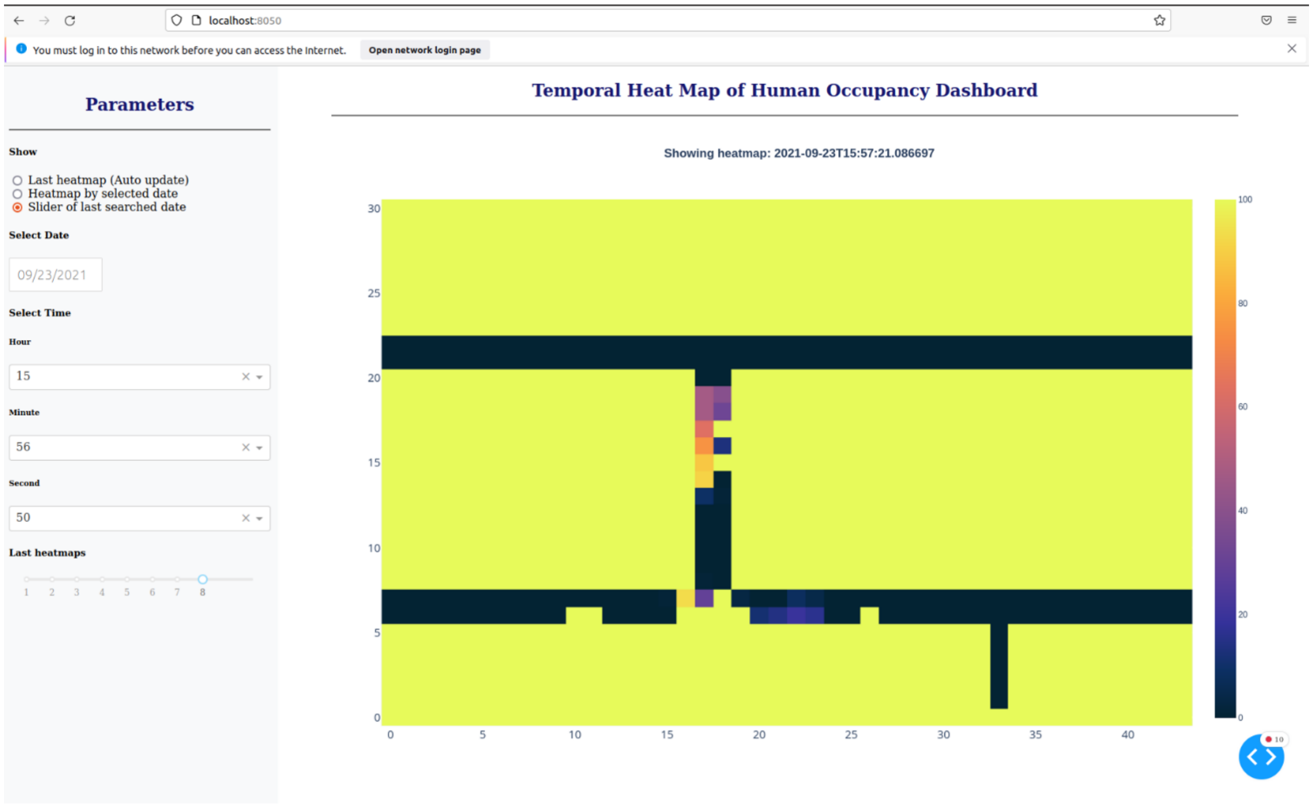
Figure 4 illustrates the DevOps pipeline overview for the deployment of components in edge and cloud on premises.

1. ROS developer commits the changes to the GitHub repo
2. The commit triggers the GitHub action workflow and runs tests and build Docker images on passing tests. The developer gets notified when there is a build failure or test failure.
3. The Docker images are then pushed to the Docker hub by GitHub actions workflow.
4. All the components for the THMHO system are published in the Docker hub.
5. The deployment team is responsible for the operation and monitoring

6. Ansible playbooks for the deployment are prepared
7. SSH keys of the control agent is copied to the controlled agents
8. The IP address of the machines and devices are listed in the inventory. Ansible playbooks are executed to pull the published images from Docker hub to the corresponding machines and then run them.
9. The deployed containers can be monitored using portianer.io

Accessing heat map from the DB?

The best way to see the heat map stored in the database is using the UI dashboard. The dashboard can be accessed via http://localhost:8050. The dashboard lets one search the data of a preferred date and time and inspect the last 10 heat maps.


localhost:8050

You must log in to this network before you can access the Internet. Open network login page

Parameters

Show

- Last heatmap (Auto update)
- Heatmap by selected date
- [✓ Slider of last searched date]

Select Date: 09/23/2021

Select Time

Hour: 15

Minute: 56

Second: 50

Last heatmaps: 8

Temporal Heat Map of Human Occupancy Dashboard

Showing heatmap: 2021-09-23T15:57:21.086697

Figure 5: UI dashboard for THMHO

One can also query heat map information from API endpoint but the result will be a *json* payload, which may not provide any meaningful information for visual inspection. This method is suggested only for consuming the heat map data for other applications. Additionally an API client can be implemented in any programming language to consume this data directly for further data processing.

```
curl –v http://localhost:5000/{"date": {"$gte": lower_time interval, "$lte": high_time_interval}}
```