AME Raptor

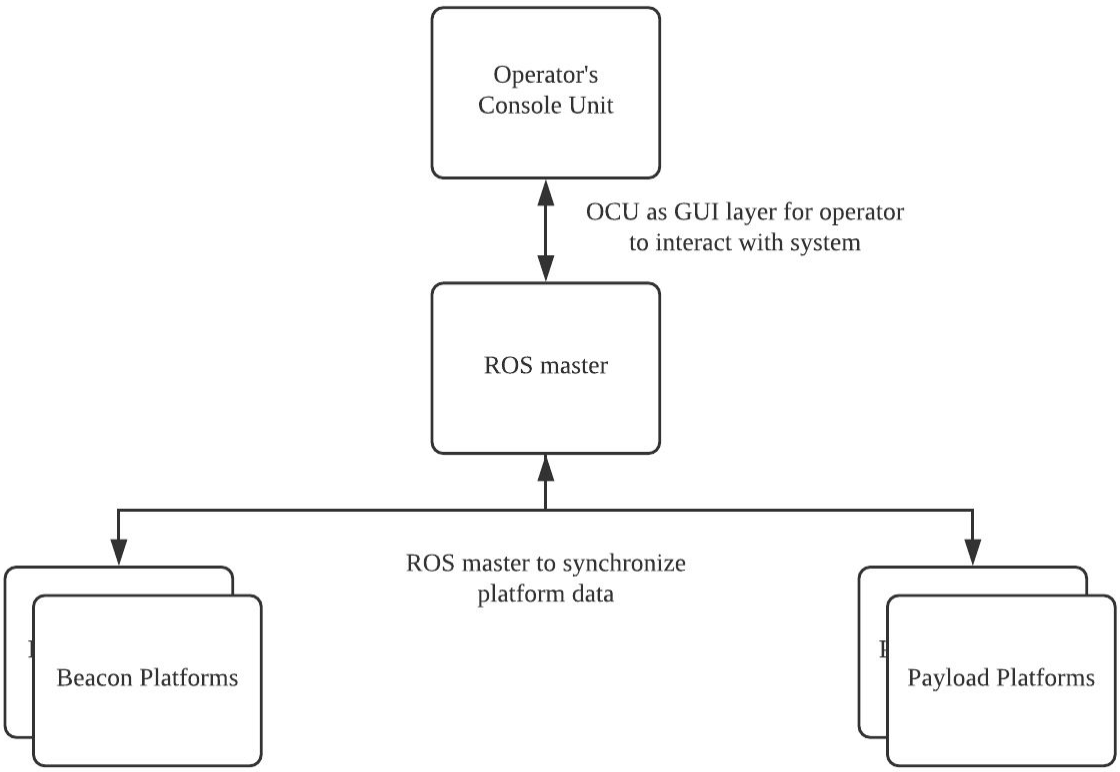
Final Report

Contents

Introduction

This report documents the specifications and setup for the Raptor system, as well as the performance and test results for the aforementioned system. The constructed system was designed and developed closely to specifications given (Refer to appendix for details).

System Architecture

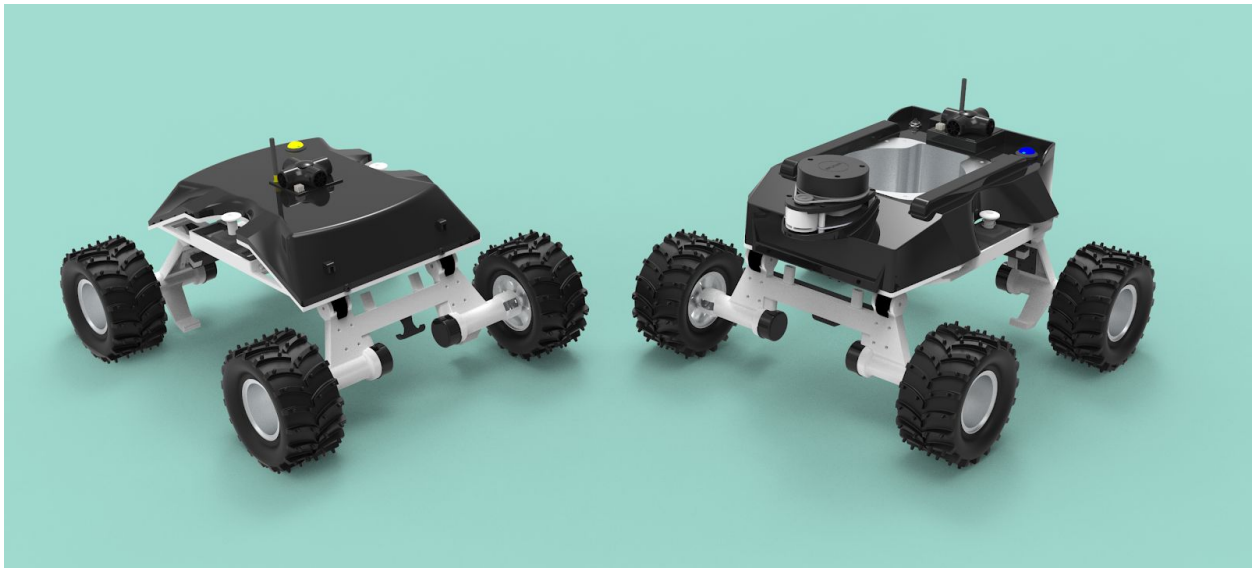


The constructed Raptor system consists of the following:

- Payload platform, the robot which carries the payload and executes formation orders
- Beacon platform, robot which carries stationary beacons for payload platform localization
- ROS master, main controller in which all information received, processed, and sent
- Operator’s Console Unit, graphical user interface for the operator to command platforms.

Mechanical Design

This section describes the specifications of the developed beacon and payload platforms.



Beacon And Payload Platforms

Design Considerations:

1. The Design of payload Platform is started by packaging the components around the payload kept at centre and as low as possible.
2. The product is supposed to be compact before and after dispatching in the field. Thus, Wheel Tucking in Mechanism was implemented in both the versions.
3. The Lidar and Beacon are placed on the cover at a position suitable for both of them not benefiting either one of them.
4. The placement of Battery,Controller and drivers are mainly based on interference with other dynamic parts in the robot and wire harnessing from front to rear of the robot as the payload is placed in the middle.
5. The Beacon platform without Lidar was advantageous to keep the Beacon at center. Excluding That , The design of the beacon robot is a replica of the Payload platform.
6. The tuck-in and out mechanism for the limbs with big wheels are designed according to the dimension constraint and kinematics of the sub assembly with switches and spring-return mechanism.

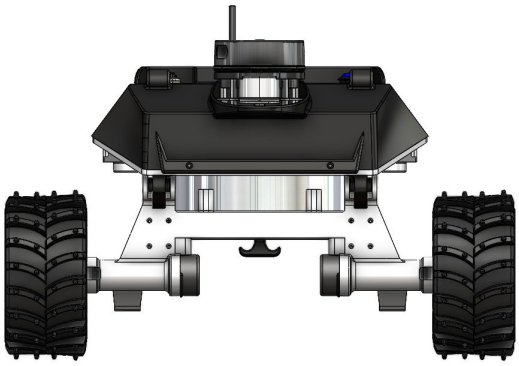
Specification	**Target**	**Achieved**
Platform weight	<=1.2kg	~2.45kg
Payload	Up to 1.6kg	Up to 1.6kg
Stowed dimensions	400x400x120mm	390x350x200mm (payload)
Environmental	Sealed to IP65	3D printed nylon for prototyping

Ground clearance	100mm	98mm stowed, 150mm unstowed

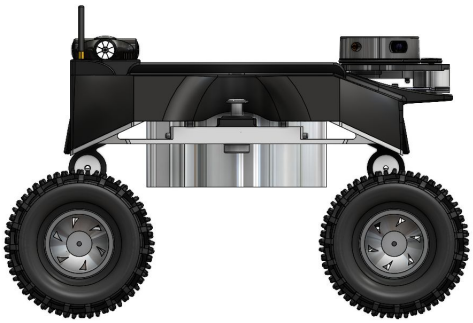
Platform Specifications (Both Payload and Beacon platforms)	
Maximum linear velocity	0.22 m/s
Maximum angular velocity	0.85 rad/s
Maximum gradient	20, 25
Maximum side gradient	18, 20
Traverse terrain	Tested on short grassland

Payload Platform

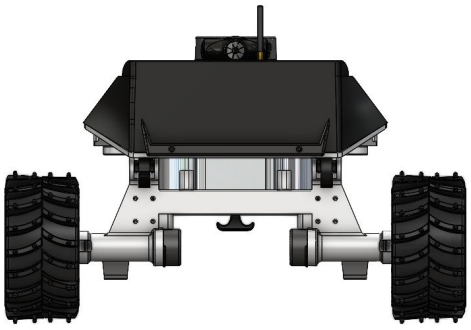
Tuck in Orientation



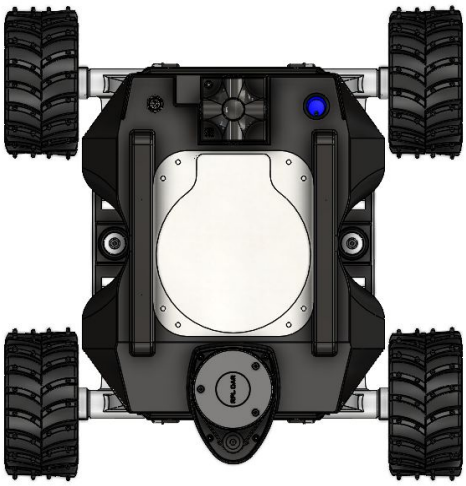
Front



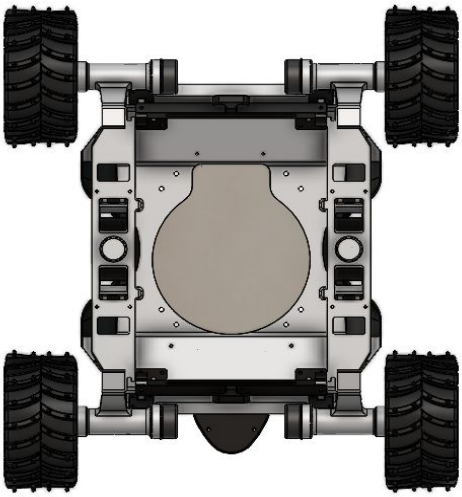
Right



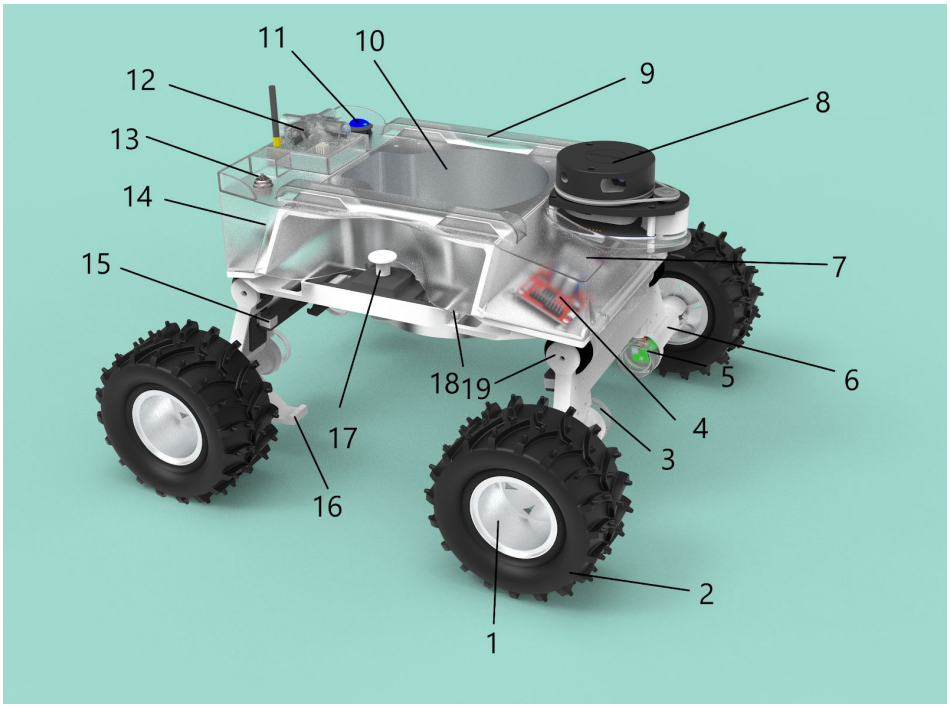
Back



Top



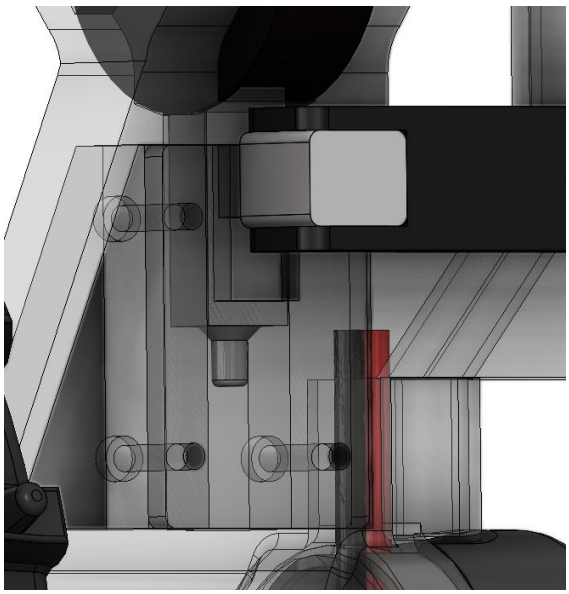
Bottom



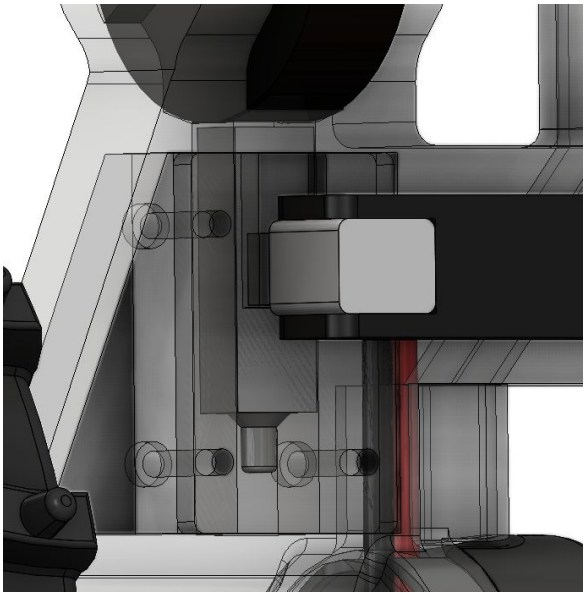
Payload Platform Schematic

1- Rim with Metal hub
2- Rubber Tyre
3- Leg with encoder Cap
4- Motor Driver
5- Encoder
6- Motor inside the leg Boss
7- LiPo Battery
8- RPLidar

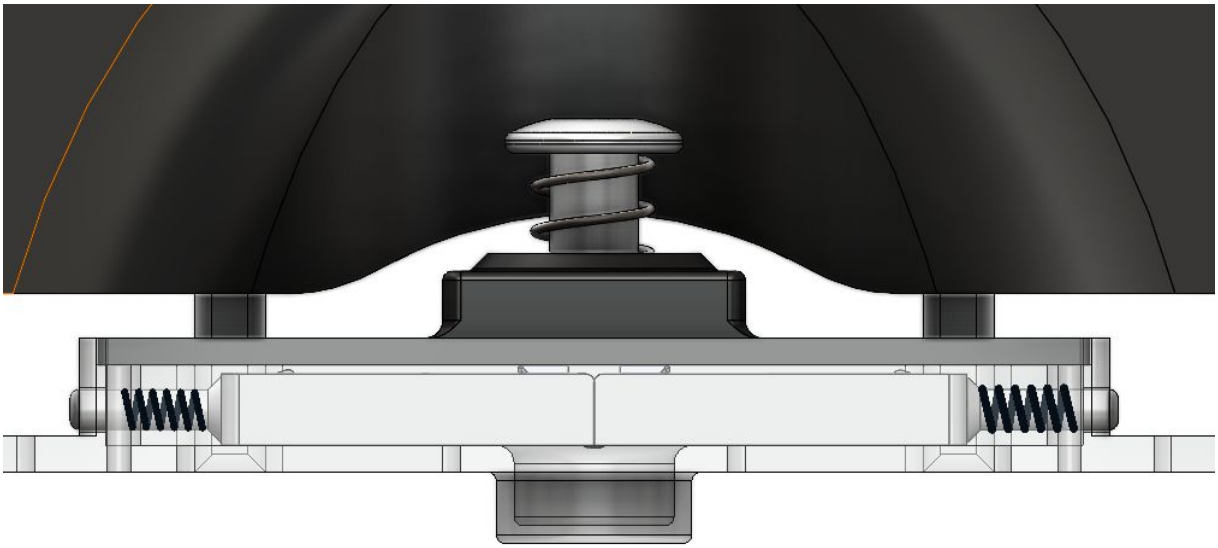
9- Top Cover with Harnessing provision
10- Payload Container
11- Power Switch
12- Marvelmind Beacon
13- 12 pin Connector for payload
14- Raspberry Pi controller
15- Lever for tuck in mechanism
16- Latch attached to leg
17- Spring Operated Push Button Release for the leg latch
18- Spring operated Latch Lock plate slider
19- Torsion spring operated revolute joint with leg holder



Lever Lock



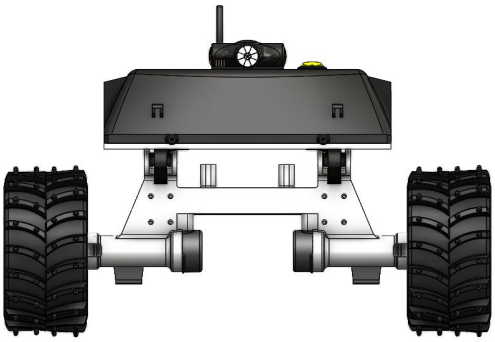
Lever Release



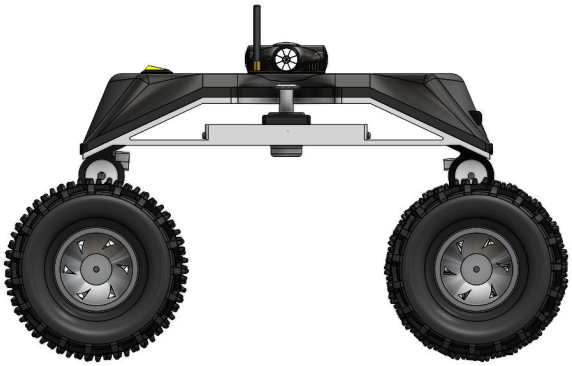
Leg Latch And Push release mechanism

Beacon Platform

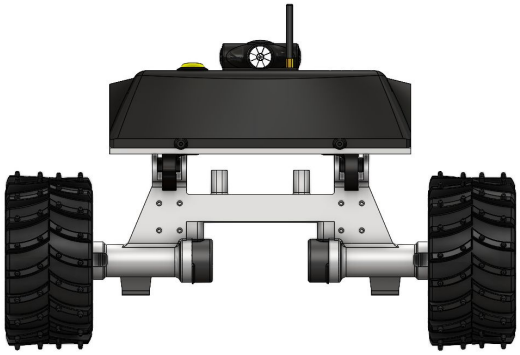
Tuck in Orientation



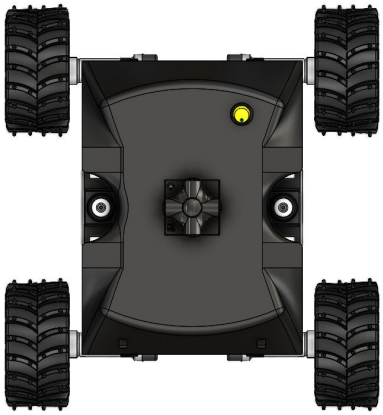
Front



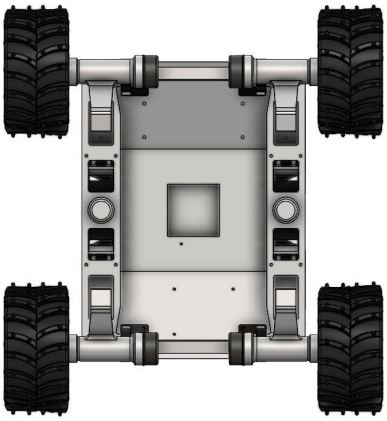
Right



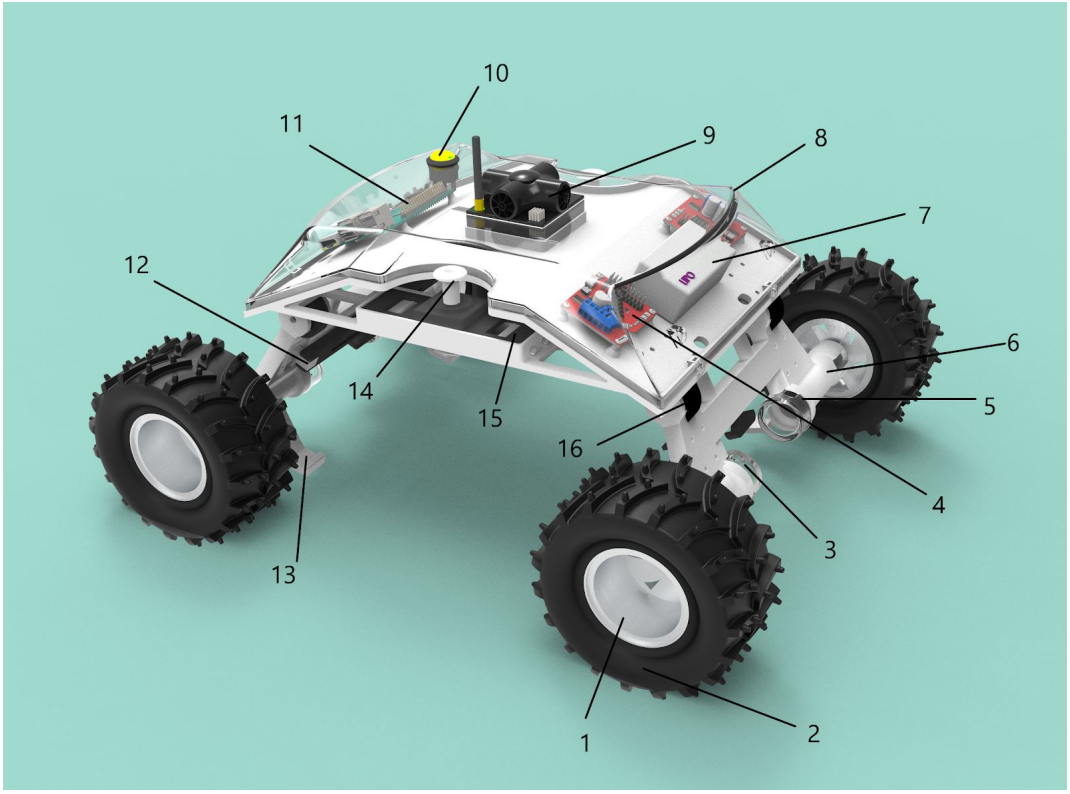
Back



Top

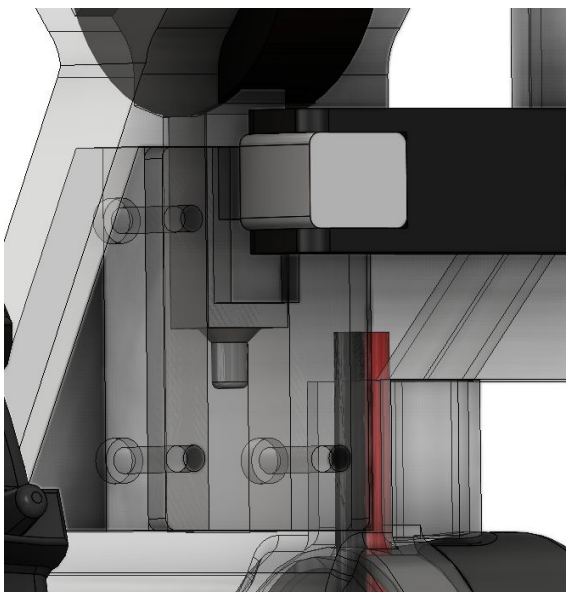


Bottom

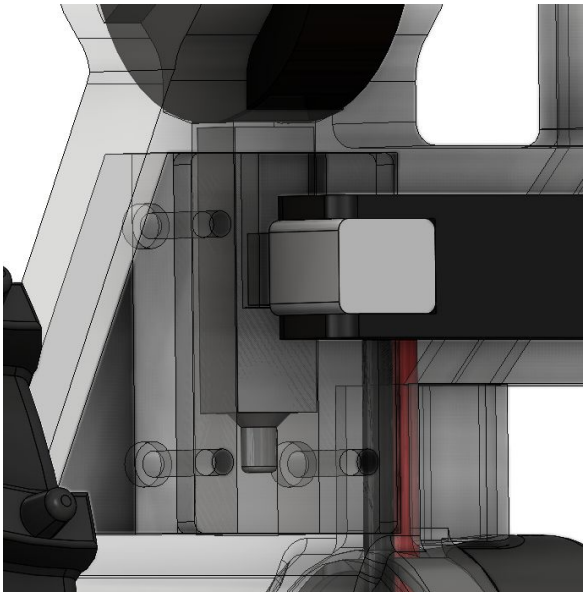


Beacon Platform Schematic

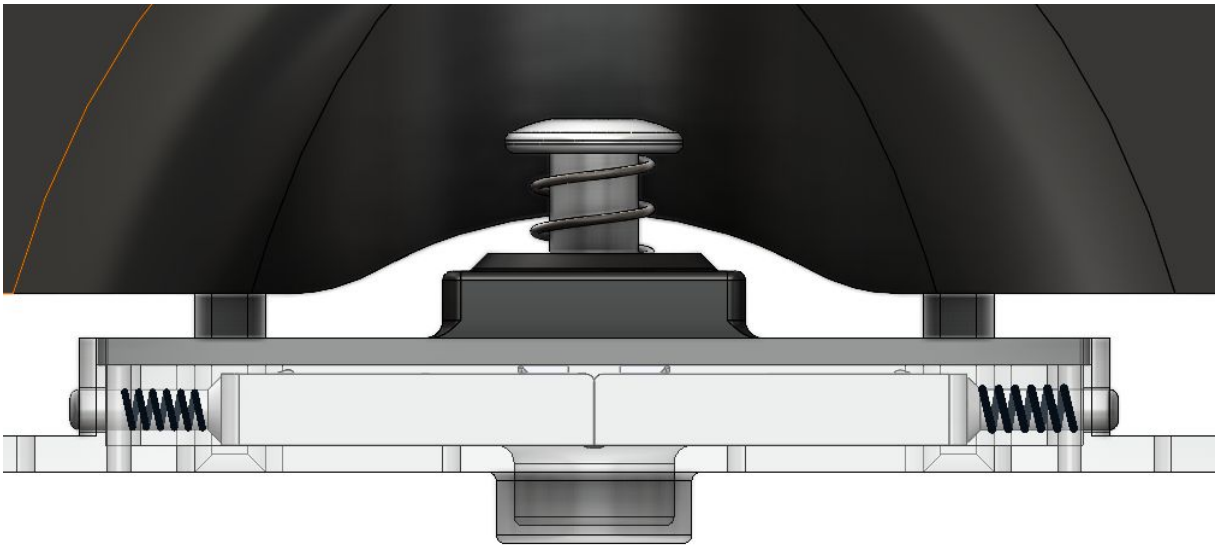
1- Rim with Metal hub
2- Rubber Tyre
3- Leg with encoder Cap
4- Motor Driver
5- Encoder
6- Motor inside the leg Boss
7- LiPo Battery
8- Top Cover with Harnessing provision
9- Marvelmind Beacon
10- Power Switch
11- Raspberry Pi controller
12- Lever for tuck in mechanism
13- Latch attached to leg
14- Spring Operated Push Button Release for the leg latch
15- Spring operated Latch Lock plate slider
16- Torsion spring operated revolute joint with leg holder



Lever Lock

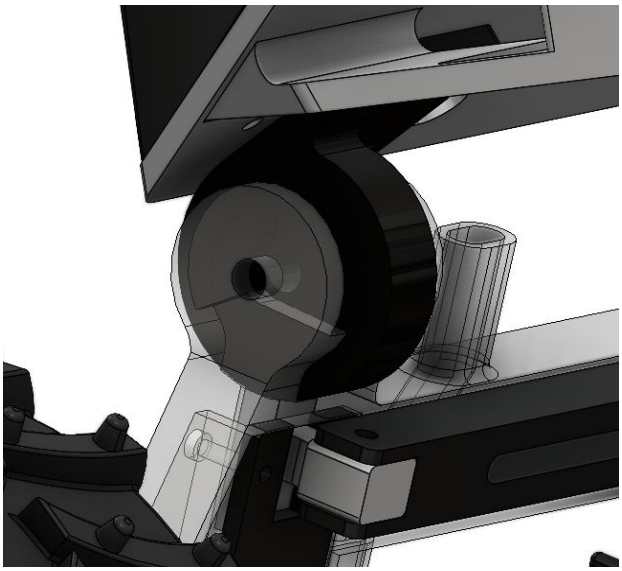


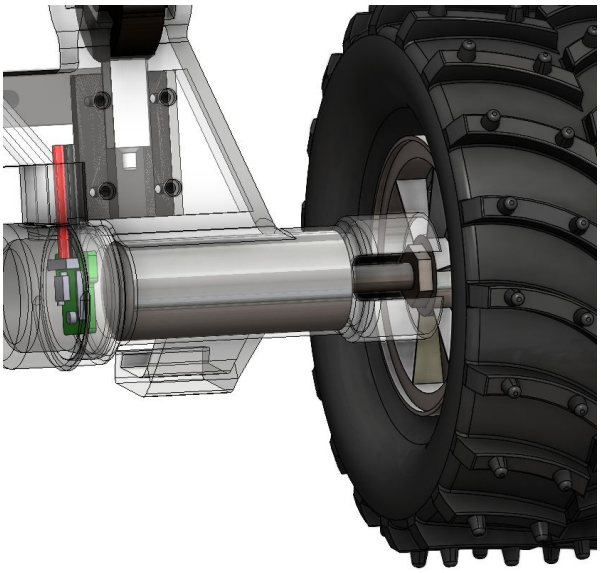
Lever Release

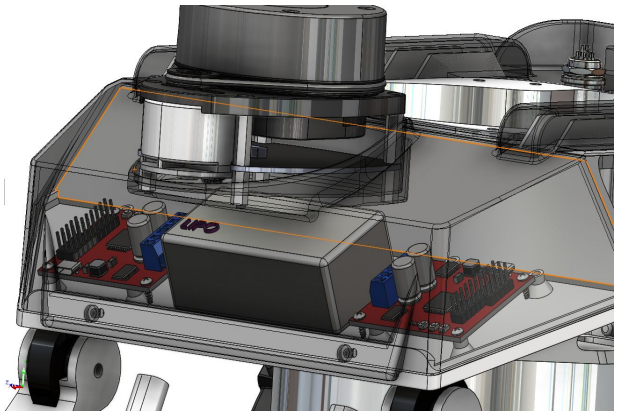


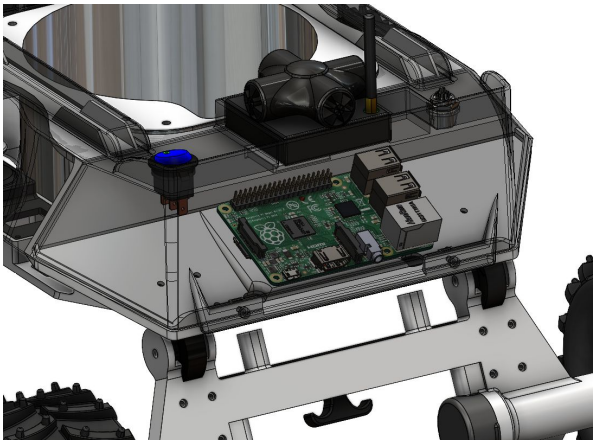
Leg Latch And Push release mechanism

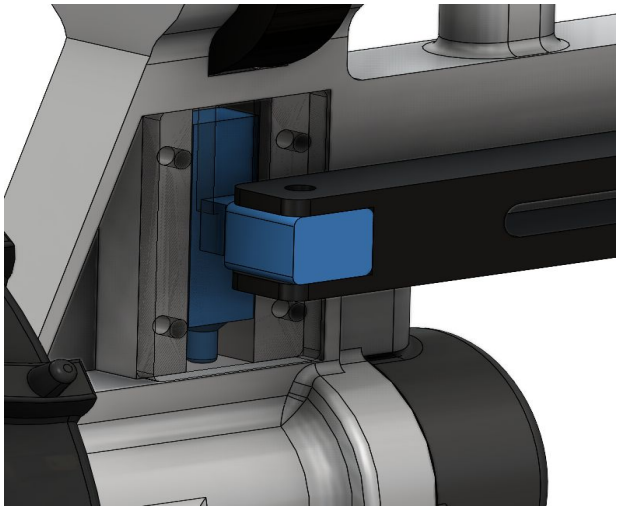
Assembly

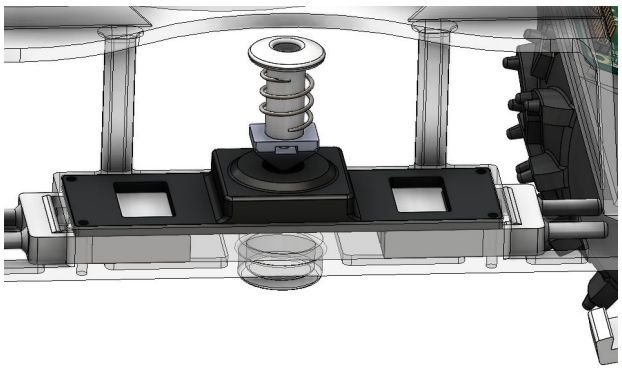












Autonomy

This section describes the autonomous layout of the raptor system. Communication between all the platforms are through ROS using the Wi-Fi medium. The central workstation runs as the ROS master and the payload and beacon platforms are connected to master as ROS slaves.

The autonomy mainly consists of two sections i.e. Localization of platforms and Collision Detection and Navigation of platforms. The platform in both sections strictly refers to the payload platform only where autonomy is required for point to point movement.

Localization

In order to navigate robots autonomously in a given workspace, the localization of the robot is very important. Due to the environment, cost and accuracy challenges, traditional localizing sensors or methods like GPS, SLAM using Lidar were not used to localize. This led to use of new technology, UWB Super Outdoor Beacons by Marvelmind, which is kind of an indoor GPS, is used to get the location of the robot. This technology can provide an accuracy of upto 2cm, provided the optimal conditions.

UWB beacon, Absolute IMU and wheel odometry of the robot datas are combined together to localize the robot.

How does Marvelmind Beacons Work?

This technology is like an indoor GPS system. At Least 3 stationary beacons are used to form a map in the given area. These 3 beacons are placed at the different corners of the work space. Then robots which need to be operated in this workspace need to carry a mobile beacon. As the mobile beacon moves the location of the beacon updates with respect to the map formed the stationary beacon, which indirectly gives the location of the robot in the map formed.

To localize the system, Marvelmind has developed two architectures in the system. Namely, Inverse Architecture and Non Inverse Architecture.

Non Inverse Architecture (NIA)

In this architecture all the stationary beacons act as the receivers and all the mobile beacons act as the transmitters .All the receivers i.e. stationary beacons are operated under the same frequency.As the number of mobile beacons increase, update rate of location of the mobile beacon decreases. If only one mobile beacon is added, the maximum location update rate is up to 16Hz. If an extra mobile beacon is added to the system then the update is divided by 2 for each mobile beacon which is 8 hz.

Inverse Architecture (IA)

In this architecture all the stationary beacons act as transmitters and all the mobile beacons act as the receivers. All the stationary beacons are operated under different frequencies. The

update rate for each mobile beacon is the same in this architecture. But the setup is a bit more complex this way.

The location update rate should be decent enough for the autonomous navigation. To facilitate this IA has been configured in the raptor project.

Initial Orientation Estimation

The IMU data is not directly fused with the beacon data as the both data informations are in different frames and the datas cannot be transformed. So there is a need to find the transformation between frames in order to find the orientation of the robot in the beacon frame. In order to do i.e. localize the robot, a novel self developed method called “Initial Orientation Estimation” is used. This method finds the initial orientation angle of the robot in the beacon frame and then on, it gets updated depending on the robot movement. So this initial orientation estimation is the key step for each raptor in the robot workspace to localize itself properly.

Finding the Initial Orientation Angle

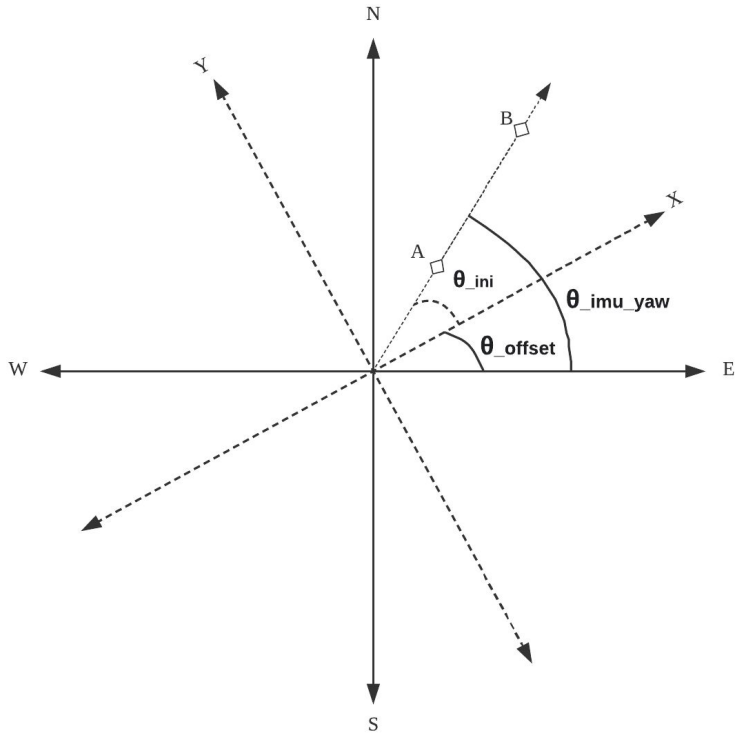
The initial orientation angle is found solely using beacon generated coordinates with the help of basic slope and linear equations.
The robot is initially moved forward for 3 seconds. During this process the location as coordinate points namely A & B generated by beacons are recorded before and after robot forward motion. These two coordinate points are used in finding the robot heading in the beacon frame using the slope formed by two points A and B.

In the below diagram, The X-Y coordinate axes and cardinal directions represent the beacon frame and the IMU frame respectively. These two frames may not be oriented in the same way and let us consider that the beacon frame is **θ_offset** times rotated about geographical axes.

Initially, when the robot is moved forward, Consider it moved from point A to B. Let A be (x1,y1) and B be (x2,y2). Then slope of AB is given by

$$tan(\theta_{ini}) = = (y2 - y1) / (x2 - x1)$$
$$\theta_{ini} = arctan((y2 - y1) / (x2 - x1))$$

Where θ_{ini} is the initial Orientation Angle.



Calculate the Offset Angle

Using the Initial Orientation Angle and the yaw generated from the absolute IMU, the relation between the beacon frame and the IMU frame is calculated by finding the rotation angle between the frames.

So cleary from the diagram,

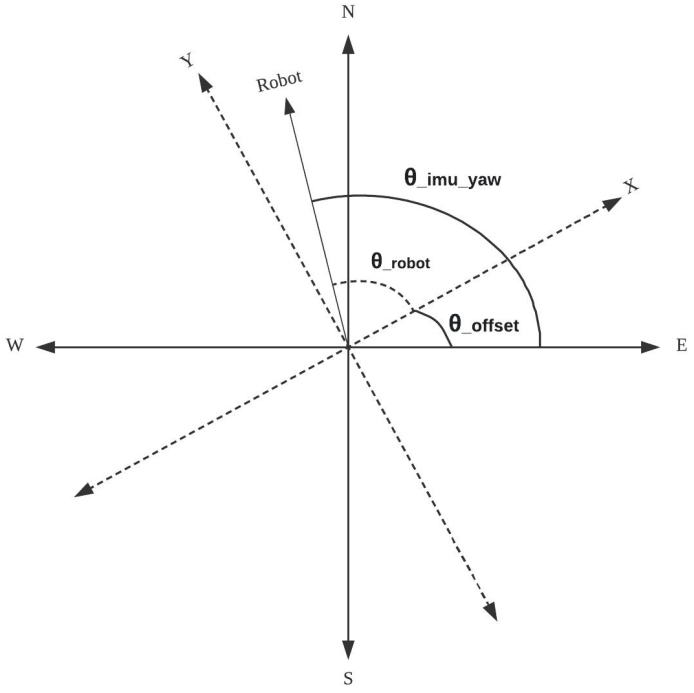
$$\theta_{imu\ yaw} = \theta_{ini} + \theta_{offset}$$
$$\theta_{offset} = \theta_{imu\ yaw} - \theta_{ini}$$

And the θ_{offset} is calculated.

Heading Angle of robot in beacon frame

From now on when the robot is moved, the heading of the robot should be calculated in the beacon frame. So let us consider the below diagram where the arrow with robot index represents the orientation of the robot.

The heading angle, **θ_robot** is calculated by using the is **θ_imu_yaw** and **θ_offset** from the previous step.



From diagram,

$$\theta_{imu\ yaw} = \theta_{robot} + \theta_{offset}$$
$$\theta_{robot} = \theta_{imu\ yaw} - \theta_{offset}$$

Where, **θ_robot** is the robot heading angle in the beacon frame,
θ_imu_yaw is the robot heading angle given by IMU cardinal directions.

Sensor and Odometry fusion

Data provided by the beacons and the robot heading angle generated by the Orientation Estimation Method are fused to form an odometry. Before fusing data together, everything is properly converted to ROS right handed frames. This data is subjected to jumps as the beacon coordinates is a discrete kind of data.
Along with the beacon odom, wheel odometry is generated from the wheel encoders. This odometry is continuous but drifts over time. So to avoid the discrete jumps, these two odometry are fused through Robot Localization ROS package for the correct localization of the robot.

This final odometry is used in navigating the robot to the desired point.

Collision Detection and Navigation

Move_base plugin where it provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base.The move_base node links together a global and local planner to accomplish its global navigation task.
Running the move_base node on a robot that is properly configured results in a robot that will attempt to achieve a goal pose with its base to within a user-specified tolerance.

For more information on move_base and navigation stack, please refer to move_base - ROS Wiki.

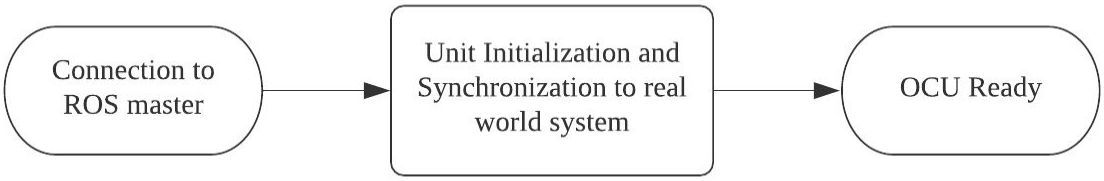
A Low cost RPLidar A1 is mounted on top of each robot to scan the surroundings as dynamic obstacles during its navigation. So each robot sees another robot as an obstacle and tries to avoid collision while reaching its goal.

Move_base can be configured to use with different planners. It is up to the user choice for the planner to be configured. Each raptor is configured with DWA local planner and NavFn global planner. The parameters are fine tuned to navigate from point A to B with the ability to avoid obstacles.

Operator’s Console Unit (OCU)

The Operator’s Console Unit (OCU) is the main graphical interface to operate the system, built using Unity 2019.4, a cross-platform engine for developing graphical programs. The OCU is required to be run on a workstation connected to the same network as the ROS master.

General Program Flow



General Program Flowchart

Connection to ROS Master

On first startup, the OCU will attempt connection to the ROS master automatically. Upon connection failure, it will prompt the operator to enter the IP address of ROS master and reconnect.

System initialization

Once connected, the system will enter the initialization phase where payload platforms are initialized and displayed red on screen while the connection to beacon and payload platforms are being established.

OCU Ready

After a few seconds, the connection should be resolved which will turn active payload platforms displayed blue and reflect its real position. This indicates that the OCU is ready to be used by the operator.

OCU to ROS Master Communications

The RosSharp C# library is used in development to allow communication with ROS master via Rosbridge protocol. The main ROS master connection code is located at RosConnector/RaptorConnector.cs.

Through this, the OCU receives data and sends movement commands to and from ROS master. Table below shows the various topics connected to the OCU via Rosbridge protocol.

OCU Subscribed Topics	
raptor<raptorNo>/odometry/filtered (nav_msgs/Odometry)	Odometry data of payload platform
bp_gps/fix (sensor_msgs/NavSatFix)	GPS data
raptor<raptorNo>/ini_ori_est_angle (std_msgs/float64)	True North Angle offset from North of payload platform odometry
OCU Published Topics	
Move base	See next section for breakdown

With exception of the camera video feed, all data is connected via Rosbridge protocol. Reason being that there was no need to integrate camera data into the ROS system for this scope.

Point to Point Movement

This refers specifically to the function for payload platforms to move to a position given by the operator. The component involved in this process is the ROS move base action library. This library consists of several topics to handle input and feedback (See table below).

Action Subscribed Topics	
move_base/goal (move_base_msgs/MoveBaseActionGoal)	A goal for move_base to pursue in the world.
move_base/cancel (actionlib_msgs/GoalID)	A request to cancel a specific goal.
Action Published Topics	
move_base/feedback (move_base_msgs/MoveBaseActionFeedback)	Feedback contains the current position of the base in the world.
move_base/status (actionlib_msgs/GoalStatusArray)	Provides status information on the goals that are sent to the move_base action.

move_base/result (move_base_msgs/MoveBaseActionResult)	Result is empty for the move_base action.

Table of Move Base Topics (Retrieved from move_base - ROS Wiki)

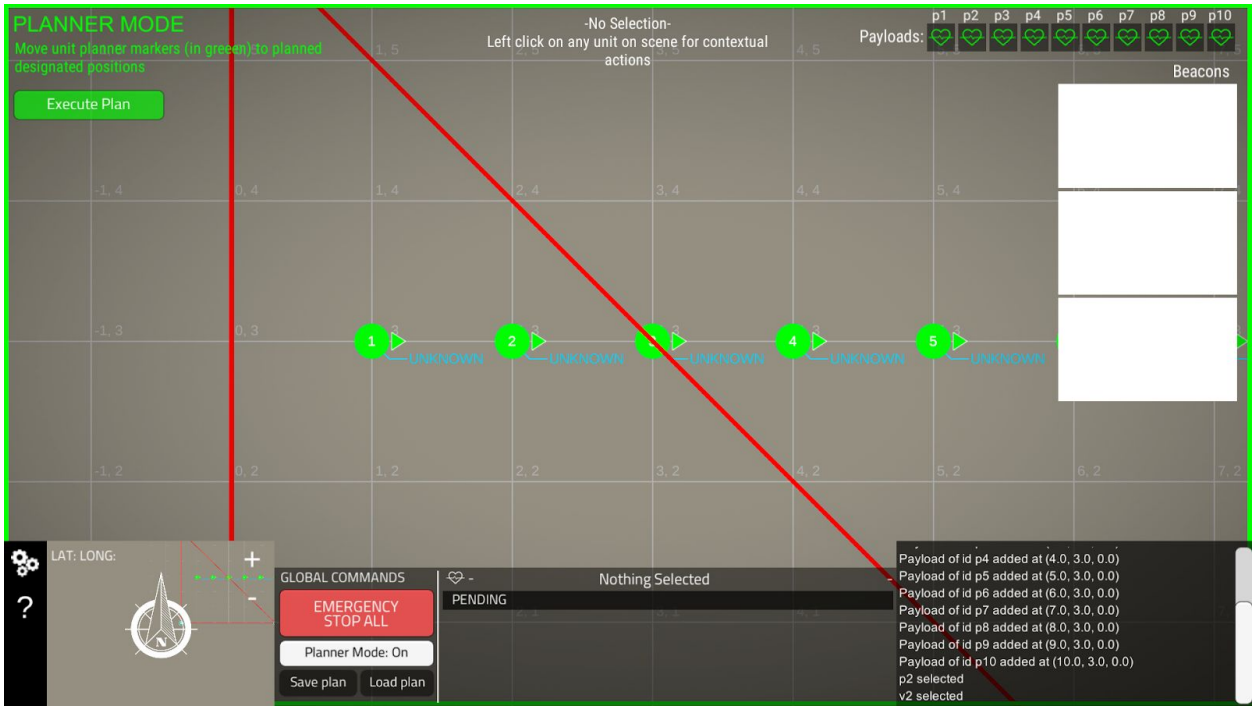
Utilizing the move base action library, point to formation movement was implemented by first generating the points for formation based, then processing point to point movement for multiple units serially. Point generation done is dependent on use cases which are expanded on in the next section.

Formation Point Generation

This section describes the formation generation functionality in the defined specifications, implemented on the OCU layer of the system rather than hardware layer to allow more control and customization over formation at the operator's end.

Self-forming

This allows the system to self deploy payload platforms without further intervention from the operator from a single operator's command.



This function is achieved by a planner mode shown above which allows the operator to pre-plan platform positions ahead of time, and then execute the movement order when desired. The planner mode can be accessed by the button under Global Commands in the OCU or automatically from the load formation button if there is already a saved formation plan. When planner mode is active, planner units (green units shown) representing the objective of the

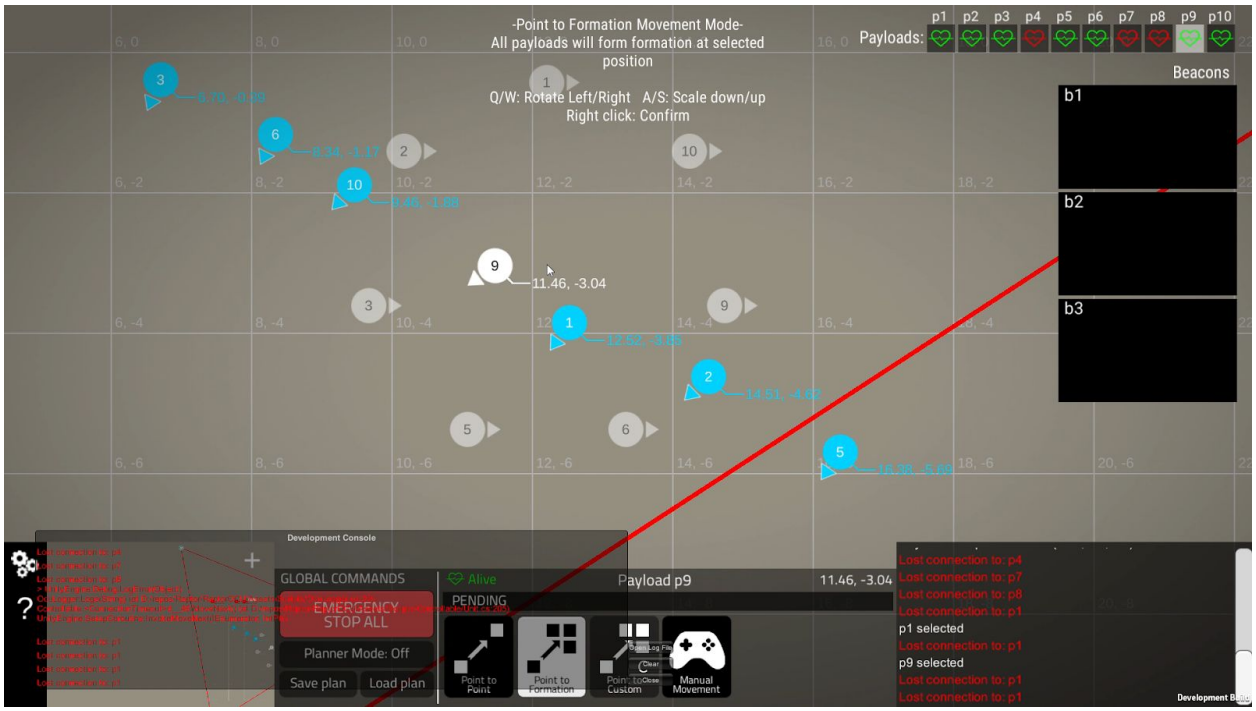
payload platforms can be manipulated to designate new objectives without affecting any of the platforms in the real world.

Once the desired formation is achieved, the operator is able to save the all positions as a formation plan for later use or execute the plan which will move all units to the projected planner unit positions.

Self-healing

The OCU has a feature to detect gaps or uneven spacing due to out-of-position, disabled or offline platforms. The system automatically redistributes the remaining platforms evenly without further operator intervention.

A mathematical approach was used for this feature, where only a single specific use case of a default circle formation was used to demonstrate this functionality.



Point to formation mode: semi-transparent units indicate for goal selection in default circle formation

Upon a circle formation execution, the projected positions will be highlighted in transparent yellow. Should any payload platforms be lost during the formation, the projected positions will reform into a circle formation with the remaining platforms, changing their objective automatically.

5 units executing circle formation

A unit lost during the process and a new goal is formed for 4 units automatically

To implement new formations with this functionality, a mathematical representation for the perimeter of desired formation needs to be derived to be able to evenly divide the perimeter to distribute payload platforms

Offline platforms are detected by a simple timeout function from payload odometry; in which if there is no response from the payload platform for a while, it is deemed lost and offline. This altogether makes up the self-healing function.

Pattern shift

The formation shape or pattern of platforms demonstrates the ability to shift its position, relative to the operator's command from the OCU with minimal intervention from the operator; with the Operator's command being limited to a combination of rotation and offset from the original position of the pattern.

As a prerequisite, the operator is required to have a formation plan saved in the workstation beforehand. The plan can be created from planner mode from the same procedure as described in the Self-forming functionality. An important thing to note when creating a plan for this function, custom formation mode will place the units relative to the mouse cursor position using the first unit (payload platform 1) as a pivot; as well as for handling rotations.

On selecting the Custom Formation button of a selected payload, this prompts the operator to load an existing plan. After that, the operator is able to rotate and position formation as desired with unit 1 as a pivot.

Known Issues

- Beacon platform position cannot be accessed due to a Marvelmind software issue. As a result the beacon platform positions shown on the OCU does not reflect nor update according to setup. For more information, please refer to Marvelmind Beacon Issue Email Chain in Appendix.
- Upon execution of movement commands, the payload platforms may not move for a while due to path planner overhead in the platform
- The payload platform may refuse to move as well due to inability to move base to find a path. This will be indicated by the status of the payload which corresponds to the move base status. In this case the operator should manually operate the platform to a collision free path before attempting any point to point movement functions.

System Tests and Results

A general test was designed for the system to identify problems and determine system collision avoidance capabilities.

Test Area

The test was conducted in a short grassfield. Stationary beacons (just the beacons without the robot) were used instead of beacon platforms to test the effect of height placement of beacon on system performance.

Variables

Manipulated variables:

- Proximity between payload platforms
- Number of payload platforms
- Height of stationary beacon platform

These variables are constant during runtime and are manipulated for each test run.

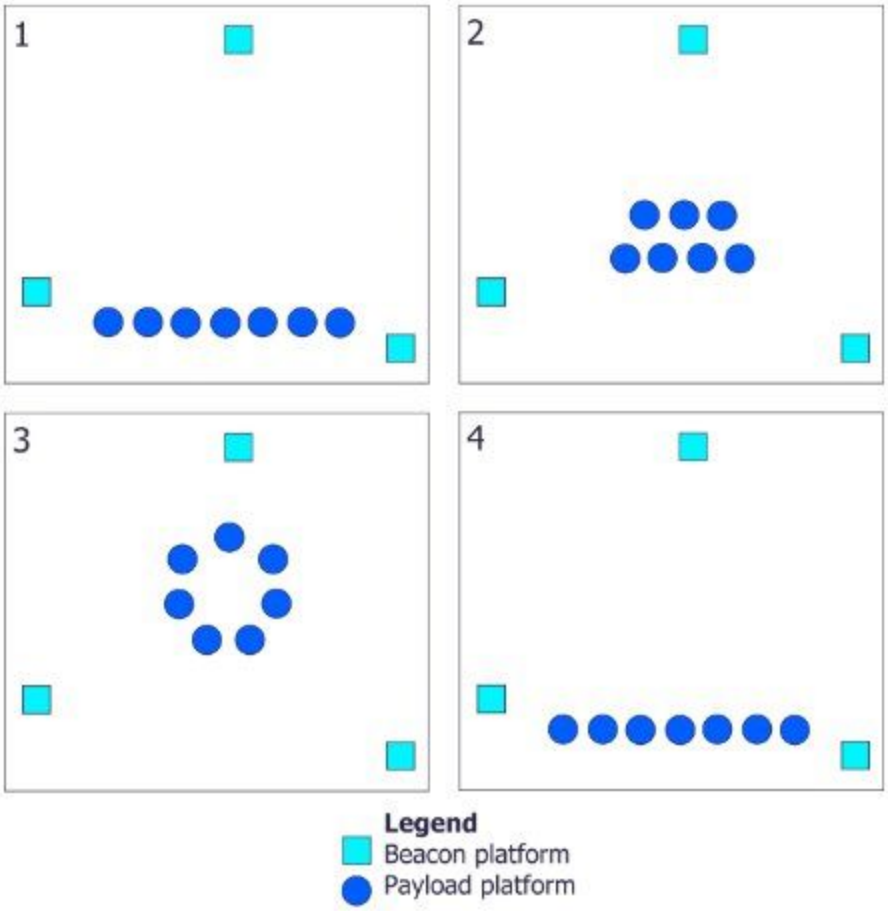
Responding variables:

- Frequency of collision during payload platform movement
- Number of response of payload platform on movement order

These variables were taken note of to determine system performance

Procedure

1. Startup all payload platforms in a straight line
2. Form a 2 row formation in forward
3. Form a circle formation forward
4. Return all payload platforms to a straight line



Test Performance Results

The system is able to achieve acceptable performance in a setup of 6 payload platforms within the operational area of 3 stationary beacons placed on height about 2m overseeing the field, and with a distance proximity between platforms for all formations of about 2m.

Mechanical Design Issues

The prototype design was found unsuitable to maneuver on grassland, so some modifications were made on the prototype.

Modified payload platform: Blocks added on top and stowed

Legs were kept stowed when in operation as unstowed movement was not stable.
Due to the position of the LIDAR sensor, the platforms had trouble detecting one another. Blocks were added to the sides to allow other payload platforms to detect itself. This helped mitigate the problem slightly, but they were still not able to detect one another at certain orientations.

Stationary beacon positioned on a tree 2m above ground

Beacon platforms were not used in the final setup as the Marvelmind beacons were required to be high enough to be in line of sight for all payload platforms. The beacons were instead placed on environment trees or poles at about 2m height above ground.

Latency issues

It is discovered that upon scaling up the number of platforms, update time from ROS topics became observably slower indicated by slower updates and more timeouts from the OCU.

As the number of payload platforms were more than 6, the latency increased to the extent some payload platforms were inoperable through move base. This is likely due to ROS unable to process too many topics in a single system.

Localization Issues

In some of the test runs, the map positions of the platforms generated by the beacons jitters variably to significant levels which severely affected performance of the navigation. This is suspected to be due to external RF interference from the outdoor environment or a defective beacon.

Localization of the robot was adequate for autonomy to perform, due to the following:

- Marvelmind Beacons together with the onboard sensors (LIDAR, IMU) were not enough for precise and smooth localization. It can be observed that the positions jumped radically at times
- In outdoors, due to the variable environment, lack of features and uneven terrain, low-cost LIDAR mapping and robot wheel odometry of the environment is not possible. As a result, localization was solely dependant on the beacons which was not enough to provide reliable localization
- As the number of payload platforms were increased, the frequency of position jumps increased in the outdoor environment. This was likely due to interference with one another

Navigation and Collision Avoidance issues

As all the aforementioned issues; namely the localization and LIDAR detection, play a major role in the navigation stack of the platform, the platform is unable to find paths at times which can cause the payload platform to abort its move goal. Parameterization was attempted but without an accurate means of localization, this could not be solved.

Conclusion

In conclusion, the aforementioned were key challenges discovered in this prototype. The following are viable topics for future research to further develop this system:

- Processor options and embedded systems
  ROS architecture exposes a lot of topics which are redundant to other platforms. The increasing number of topics is another possible cause for latency issues so some topics could be embedded to be platform specific only and not exposed to others.
- Sensor options and provision for more sensors
  The sensors onboard the platform now are insufficient to provide accurate localisation and collision detection. More sensors and qualitative options need to be explored.
- Localization techniques
  Alternative localization techniques need to be explored to operate the system in an open variable environment.

Appendix

Raptor Design Document

TECHNICAL REQUIREMENTS FOR PROJECT RAPTOR TO SUTD

Please clarify with AME PoC (homc@stengg.com) if there are any queries or clarifications.

1. Physical requirements

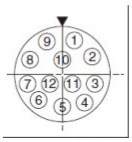
a. Platform weight: <=1.2kg
b. Payload(per platform): Up to 1.6kg
c. Stowed dimensions, platform: 400x400x120mm
d. Environmental : Platform should be sealed to IP65.
e. Ground clearance : 100mm

2. Interfaces between Payload and Platform vehicle

a. The physical interface between the platform and payload is as specified in attached drawing(s) in Annex A.
b. CG position of payload can be assumed to be within +/- 20mm of the top of the platform shell cover.

3. Electrical Interfaces

a. The Electrical interface between Payload and Platform is defined by a single 12 pin Hirose socket connector (Front panel mount, receptacle, female pin, mfg part no. LF10WBR-12S with Pin assignments as indicated below.



Pin Arrangement (Extracted from Manufacturer's catalogue)

Pin No	Assignment	Remarks / Comments
1	12VDC	2A, maximum peak, see note (c.i) below.
2	12VDC	2A, maximum peak, see note (c.i) below.
3	5 VDC*	200mA, maximum continuous, regulated, common source as pin 4. See note(c.ii) below.
4	5 VDC*	200mA, maximum continuous, regulated, common source as pin 3. See note(c.ii) below.
5	RESERVED	
6	GND (Common)	

7	GND (Common)	
8	GND (Common)	
9	RESERVED	
10	SIG A	Ready signal from platform controller, 5V (high), 0V (low)
11	SIG B	Go signal from platform controller, 5V (high), 0V (low)
12	RESERVED	

b. Position of connector shall be made known to AME as soon as feasible.

c. The platform is to supply the following sources of power through the electrical interface:
i) 12VDC, 4A peak draw for <=> 100mS.
ii) 3.3/5VDC, 250mA continuous for 30mins.

d. The platform shall be able to deploy without recharging within 2 weeks of a full charge.

4. NAVIGATION AND LOCALISATION

a. It is preferred that the system does not require the aid of a GNSS or surveyed point(s) for the determination of absolute positioning.

b. Should there be any requirement(s) to use an external source for absolute positioning, this shall be made known as soon as possible.

c. UWB beacons should be deployed with minimal user intervention. Manual emplacement of the beacons is not preferred, but can be consider if other system constraints are overriding. This is due to placement and setting up time constraints.

5. OTHER REQUIREMENTS

a. All beacon platforms shall have a video feed to be displayed on the Operator's Console Unit(OCU).

b. Power to all platform motors shall be disabled when the platform(s) is in position.

c. The Platforms shall have a 'heartbeat' function implemented to indicate its status on the OCU display.

d. All vehicular platforms shall have an physical emergency stop(E-stop) button or switch tied directly to its power source.

e. The OCU shall have a system wide E-stop button to stop the complete system in case of emergency.

f. A 'READY' signal (SIG A) shall be sent by the platform controller when the platform is in position.(**Reconsider**)

g. Set up time of not more than 2 minutes – Time required to set up the system, position beacons and platforms, start OCU, conduct built-in-tests, connect OCU and platforms, from standing start, 50m from edge of deployment position.

h. Deployment time of not more than 10 secs – System is assumed to be ready at end of setting up process. Time required to move all 10 platforms into deployment position from standing start at edge of road. Measured from start of platforms from stationary position to stop of platform in deployed formation position.

6. PROGRESSION TRACKING

It is proposed to track the implementation and progress of the project through the following internal milestones:

a) Interface control definition
b) Fitment and envelop check
c) Platform mobility tests
d) Localisation accuracy tests
e) Obstacle avoidance tests
f) Man Machine Interface(MMI) definition
g) Scenario Tests

SUTD can propose a simple schedule or dates to the above tasks.

INTERFACE CONTROL DEFINITION

Physical interfaces and connections between the platform and payload is finalised.

FITMENT AND ENVELOP CHECK

A fitment and envelop check shall be conducted to ensure all subsystems do not interfere physically with each other. Arrangement drawings and/or rough 3D model should be made available to ensure fitment and physical interoperability.

PLATFORM MOBILITY TESTS

A series of simple mobility tests shall be conducted to quantify the performance of the platform with 1.6kg payload, unless otherwise stated. These tests shall include:

a) Maximum velocity.
b) Maximum acceleration.
c) Maximum gradient – with payload and maximum payload.
d) Maximum side gradient - with payload and maximum payload.

e) Maximum Vertical obstacle.
f) Maximum Horizontal gap.
g) Traverse sandy / loose aggregate / muddy terrain
h) Pivot Turns or minimum turn radius .

LOCALISATION ACCURACY TEST

This test is to ascertain the positioning accuracy of the UWB and the ability of the platform to move to their assigned positions within an error of +/-5cm.

OBSTACLE AVOIDANCE TESTS

The system shall demonstrate the ability to avoid and work around unsurmountable physical obstacles.

MAN-MACHINE INTERFACE DEFINITION

Graphical documentation outlining the general appearance, functions, options, commands, symbology, indicators and system responses shall be generated, so that a clear understanding of how the system will function and react can be understood.

SCENARIO TESTS

The following tests are to demonstrate the following features of the system:

a) 'Self-forming' – The system shall be demonstrate the ability to self deploy without further intervention from the operator from a single operator's command.
It can be assumed that this will take place from the state of system set up completion.
It can be further assumed that all of the platforms are at the edge of the deployment area.

b) 'Self-healing' – The deployed pattern shall demonstrate the ability to detect gaps or uneven spacing due to out-of-position, disabled or offline platforms.
The system shall automatically redistribute the remainng platforms evenly without further operator intervention.

c) 'Pattern shift' – The formation shape or pattern of platforms shall demonstrate the ability to shift its position, relative to the operator's command from the OCU with minimal intervention from the operator.
The Operator's command being limited to a combination of rotation and offset(s) from the original position of the pattern.

Context: This is in the event that a quick adjustment of the deployed pattern necessary to ensure the most optimum coverage possible on the ground.

7. DELIVERABLES

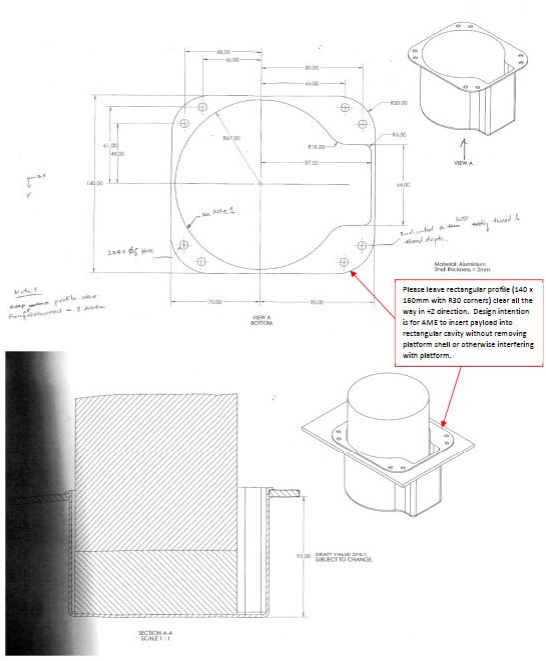
a) Review Report
b) User manual documentations
c) Ten developed Raptor platforms in working condition
d) Ten developed user interface kits
e) Software codes and documentation

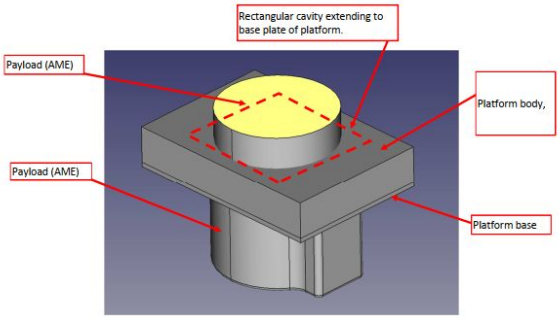
Please clarify with AME PoC (homc@stengg.com) if there are any queries or clarifications.

Ver 1.0

25/02/2020

ANNEX A: Physical Interface between payload and platform





Mechanical Design

Platform Electronic components

Component	**Model**	**Quantity for 1 unit**
DC motor	391:1 Metal Gearmotor 20Dx46L mm 12V CB with Extended Motor Shaft/3496	4
Magnetic Encoder	Magnetic Encoder Pair Kit for 20D mm Metal Gearmotors, 20 CPR, 2.7-18V	4
Motor Driver	Roboclaw 2*7A	2
Marvelmind Super Beacons Outdoor	HW-V 4.9 (without IMU)	1
Voltage Regulator 12V to 5V	D24V50F5/2851	1
Voltage Regulator (for motors) 12V to 12V	S18V20F12/2577	1
Raspberry Pi	-	1
IMU	SparkFun VR IMU Breakout - BNO080 (Qwiic)	1
Micro SD Card		1
Lipo Battery	14.4V 2800mAh UR18650ZY-4S1P-AAM battery	1
LIDAR	RPlidar A1M8	1
GPS	U-Blox C94-M8P-2	1 (for beacon platform only)
Pi camera	5MP CSI Camera Module	1 (for beacon platform only)

Payload Platform ROS files - Description

The ROS workspace called raptor_ws in “~/Home” contains all the required ROS packages.

ROS Package	Description
cmps_imu	IMU package
gps_navigation	Some required launch files present in this package
rplidar_ros	RPLidar ROS package
marvelmind_nav	Marvelmind Beacons package
laser_filters	Filter for laser scan of lidar package
raptor_2d_nav	Move Base parameters and configuration package
raptor	Payload platform teleop,wheel odom codes and launch files containing package
robot_localization	Robot localization ROS package for odometry fusion

Once turned on the payload platform a script “~/Desktop//sh_files/startup.sh” is run automatically.
This launches all the required launch files for the payload platform.

First it will launch raptor_group.launch
This launch file starts all the sensors and hardware of the robot i.e IMU, Beacons, Lidar and motor drivers. This also performs the initial orientation estimation.

Launch file or python or c++ executable	package	Description
cmps_imu_node	cmps_imu	Start IMU and publish data
hedge_rcv_bin	marvelmind_nav	Publish beacon datas
rplidar_group.launch	rplidar_ros	Publish lidar scan data
ini_orientation_estimation_group.py	raptor	Perform initial estimation for payload platform.

Second launch file is map_n_odom_group.launch

Launch file or python or c++ executable	package	Description
static_transform_publisher	tf	Publishes static transforms between various ROS frames. This is in accordance with how the real sensor frames are oriented and translated between each other.
hedge_msg_adapter_twist_msg_group	marvelmind_nav	Fuses beacon location data and IMU heading angle to form the discrete odometry.
raptor_base_controller_node_group.py	raptor	Node to drive the robot and simultaneously generate wheel odometry.

Third launch file, raptor_ekf_template_group.launch

Launch file or python or c++ executable	package	Description
raptor_ekf_template_group.launch	robot_localization	Fuses beacon odometry, wheel odometry using ROS robot localization project for payload platform localization.

Fourth launch file, move_base_2_group.launch

Launch file or python or c++ executable	package	Description
move_base	move_base	ROS launches the movebase for autonomous navigation with required robot and planner parameters.
range_filter_example_group.launch	laser_filters	Filters noise for the laser scan provided by the rplidar node for autonomous navigation.

Beacon Platform ROS files - Description

The ROS workspace called raptor_ws in “~/Home” contains all the required ROS packages.

ROS Package	Description
cmps_imu	IMU package
raptor	beacon platform teleop,wheel odom codes and launch files containing package

Once turned on the payload platform a script “~/Desktop//sh_files/startup.sh” is run automatically.
This launches all the required launch files for the beacon platform.

First it will launch beacon_group.launch

Launch file or python or c++ executable	package	Description
cmps_imu_node	cmps_imu	Start IMU and publish data
beacon_base_controller_node_group.py	raptor	Node to drive the robot and simultaneously generate wheel odometry.
static_transform_publisher	tf	Publishes static transforms between various ROS frames. This is in accordance with how the real sensor frames are oriented and translated between each other.

Second, it runs the camera node for the live stream.

file	location	Description~
main.py	~VideoServer/main.py	Runs the camera node for the live stream. Used for monitoring.

Operator's Console Unit

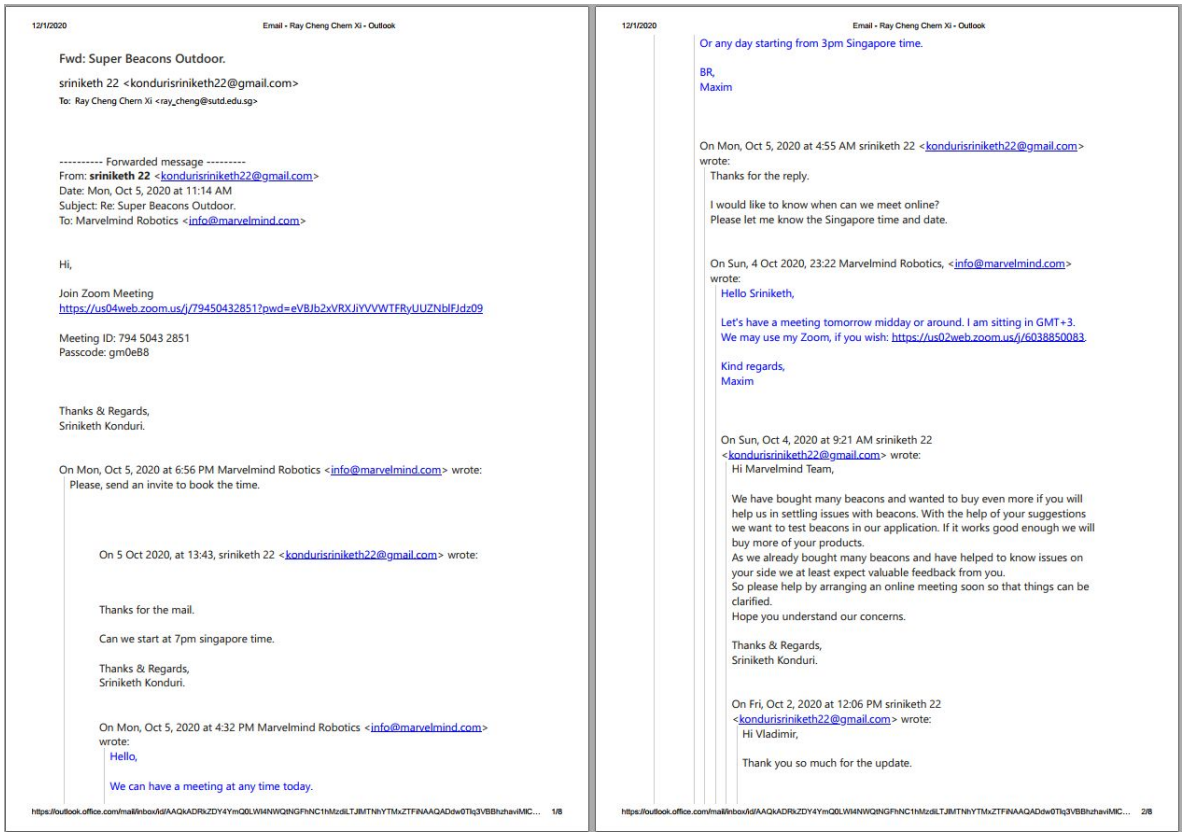
Unity C# scripts:

```
## Script Hierarchy
Hierarchy of the c sharp files
```
```
Scripts
├── Extensions
│   ├── CustomWebRequest.cs     //Web handler for video feed
│   ├── SerializableTypes.cs    //Serializable helper for data structures for save/load
│   └── Singleton.cs            //Unity Monobehaviour singleton base
├── Controllable                //contains scene unit inheritables
│   ├── Unit.cs                 //Base parent class for all units on scene
│   ├── Payload.cs              //Payload platform unit, child of Unit
│   ├── Beacon.cs               //Beacon platform unit, child of Unit
│   ├── PlannerUnit.cs          //Planner unit used in planner mode, child of Unit
│   └── PayloadDisplayItem.cs   //Payload UI display, referenced in Payload.cs
├── RosConnector                //contains additional RosSharp definition
│   ├── Actions                 //contains generated topic handlers from RosSharp for MoveBase
│   │   ├── MoveBaseActionFeedback.cs
│   │   ├── MoveBaseActionGoal.cs
│   │   ├── MoveBaseActionResult.cs
│   │   ├── MoveBaseFeedback.cs
│   │   ├── MoveBaseGoal.cs
│   │   └── MoveBaseResult.cs
│   ├── Messages                //contains additional message data structures for GPS data
│   │   ├── NatSatFix.cs
│   │   └── NavSatStatus.cs
│   ├── RaptorConnector.cs      //Modified from RosConnector.cs, optimized for Unity and Raptor usage
│   └── MoveBaseActionClient.cs //Handler for MoveBaseAction
├── OcuManager.cs               //Overall scene manager. Handles unit initialization, in scene display
├── UiManager.cs                //Unity Canvas manager. Handles GUI states.
├── OcuLogger.cs                //Logger for logger display
├── OcuLogItem.cs               //Dynamic list entry for logger display
├── CameraPan.cs                //Handles camera pan movement and grid drawing
├── Compass.cs                  //To show geographic North
├── RaptorPlanData.cs           //Data class for saving/loading formation plan files
├── WorldScaler.cs              //Static class for default scale sizes
└── PlayerPrefsConstants.cs     //Constant definition for Unity's PlayerPref variables
```

Move Base status

```
PENDING = 0,    // The goal has yet to be processed by the action server
ACTIVE = 1,     // The goal is currently being processed by the action server
PREEMPTED = 2,  // The goal received a cancel request after it started executing
                //   and has since completed its execution (Terminal State)
SUCCEEDED = 3,  // The goal was achieved successfully by the action server (Terminal State)
ABORTED = 4,    // The goal was aborted during execution by the action server due
                //    to some failure (Terminal State)
REJECTED = 5,   // The goal was rejected by the action server without being processed,
                //    because the goal was unattainable or invalid (Terminal State)
PREEMPTING = 6, // The goal received a cancel request after it started executing
                //    and has not yet completed execution
RECALLING = 7,  // The goal received a cancel request before it started executing,
                //    but the action server has not yet confirmed that the goal is canceled
RECALLED = 8,   // The goal received a cancel request before it started executing
                //    and was successfully cancelled (Terminal State)
LOST = 9        // An action client can determine that a goal is LOST. This should not be
                //    sent over the wire by an action server
```

Marvelmind Beacon Issue Email Chain



12/1/2020 Email - Ray Cheng Chern Xi - Outlook

Fwd: Super Beacons Outdoor.

sriniketh 22 <kondurisriniketh22@gmail.com>
To: Ray Cheng Chern Xi <ray_cheng@sutd.edu.sg>

---------- Forwarded message ---------
From: **sriniketh 22** <kondurisriniketh22@gmail.com>
Date: Mon, Oct 5, 2020 at 11:14 AM
Subject: Re: Super Beacons Outdoor.
To: Marvelmind Robotics <info@marvelmind.com>

Hi,

Join Zoom Meeting
https://us04web.zoom.us/j/79450432851?pwd=eVBJb2xVRXJiYVVWTFRyUUZNblFJdz09

Meeting ID: 794 5043 2851
Passcode: gm0eB8

Thanks & Regards,
Sriniketh Konduri.

On Mon, Oct 5, 2020 at 6:56 PM Marvelmind Robotics <info@marvelmind.com> wrote:
> Please, send an invite to book the time.
>
> On 5 Oct 2020, at 13:43, sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
>
> Thanks for the mail.
>
> Can we start at 7pm singapore time.
>
> Thanks & Regards,
> Sriniketh Konduri.
>
> On Mon, Oct 5, 2020 at 4:32 PM Marvelmind Robotics <info@marvelmind.com> wrote:
> > Hello,
> >
> > We can have a meeting at any time today.

https://outlook.office.com/mail/inbox/id/AAQkADRkZDY4YmQ0LWI4NWQtNGFhNC1hMzdlLTJlMTNhYTMxZTFlNAAQADdw0Tlq3VBBhzhaviMIC... 1/8

12/1/2020 Email - Ray Cheng Chern Xi - Outlook

> Or any day starting from 3pm Singapore time.
>
> BR,
> Maxim
>
> On Mon, Oct 5, 2020 at 4:55 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
> > Thanks for the reply.
> >
> > I would like to know when can we meet online?
> > Please let me know the Singapore time and date.
> >
> > On Sun, 4 Oct 2020, 23:22 Marvelmind Robotics, <info@marvelmind.com> wrote:
> > > Hello Sriniketh,
> > >
> > > Let's have a meeting tomorrow midday or around. I am sitting in GMT+3.
> > > We may use my Zoom, if you wish: https://us02web.zoom.us/j/6038850083.
> > >
> > > Kind regards,
> > > Maxim
> > >
> > > On Sun, Oct 4, 2020 at 9:21 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
> > > > Hi Marvelmind Team,
> > > >
> > > > We have bought many beacons and wanted to buy even more if you will help us in settling issues with beacons. With the help of your suggestions we want to test beacons in our application. If it works good enough we will buy more of your products.
> > > > As we already bought many beacons and have helped to know issues on your side we at least expect valuable feedback from you.
> > > > So please help by arranging an online meeting soon so that things can be clarified.
> > > > Hope you understand our concerns.
> > > >
> > > > Thanks & Regards,
> > > > Sriniketh Konduri.
> > > >
> > > > On Fri, Oct 2, 2020 at 12:06 PM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
> > > > > Hi Vladimir,
> > > > >
> > > > > Thank you so much for the update.

https://outlook.office.com/mail/inbox/id/AAQkADRkZDY4YmQ0LWI4NWQtNGFhNC1hMzdlLTJlMTNhYTMxZTFlNAAQADdw0Tlq3VBBhzhaviMIC... 2/8

12/1/2020 Email - Ray Cheng Chern Xi - Outlook

> > > > > Can you please help in arranging an online meeting with my team so that more doubts can be clarified.
> > > > > The meeting will be very helpful and important for us as you can understand our requirement for our project and can give valid suggestions to improve our system.
> > > > >
> > > > > Hope to hear from you soon.
> > > > > Thanks & Regards,
> > > > > Sriniketh Konduri.
> > > > >
> > > > > On Wed, Sep 30, 2020 at 6:28 PM Marvelmind Robotics (Vladimir) <vladimir@marvelmind.com> wrote:
> > > > > > Hello,
> > > > > >
> > > > > > We have found the issue causing empty 'beacon_pos_a', we will fix it ASAP and inform you.
> > > > > > Thank you for the feedback.
> > > > > >
> > > > > > As for IA:
> > > > > > 1. For proper automatic map build all stationary beacons should be on line of sight to each other.
> > > > > > If this is not possible for some beacons, you can specify corresponding distance manually through a table of distances.
> > > > > > 2; 3. In IA each mobile beacon should know locations of all stationary beacons on the map and some more information for tracking.
> > > > > > Modem transmits this information to mobile beacons at moment of freezing the map.
> > > > > > So if a hedgehog was woken up after freezing the map, it can contain incorrect information about the map or no map at all.
> > > > > > This state of hedgehog is shown in the dashboard as a warning ' Incorrect Map. Please Freeze Again'.
> > > > > >
> > > > > > BR,
> > > > > > Vladimir
> > > > > >
> > > > > > On Wed, Sep 30, 2020 at 6:31 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
> > > > > > > Hi,
> > > > > > >
> > > > > > > Waiting for your solution. Please try to revert back as we bought many beacons and want to find a solution to our application.
> > > > > > >
> > > > > > > Thanks & Regards,
> > > > > > > Sriniketh Konduri.
> > > > > > >
> > > > > > > On Tue, Sep 29, 2020 at 12:27 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
> > > > > > > > Hello,

https://outlook.office.com/mail/inbox/id/AAQkADRkZDY4YmQ0LWI4NWQtNGFhNC1hMzdlLTJlMTNhYTMxZTFlNAAQADdw0Tlq3VBBhzhaviMIC... 3/8

12/1/2020 Email - Ray Cheng Chern Xi - Outlook

> > > > > > Thanks for the mail.
> > > > > > Just to clarify, the above screenshots were for the NIA setup. So the update rate and etc were for the NIA not IA.
> > > > > > I just provided that screenshots because to let you know that I the beacons_pos_a was empty.
> > > > > > Below is a file attached for NIA setup. It is a simple NIA configured setup.
> > > > > > I am using 10 hedgehogs in my robotic project. So for my autonomy to work properly I need good localization from the hedgehogs. With NIA the update rate is very slow and doesn't account for good autonomy. So to have good autonomy I wanted to set up using IA.
> > > > > >
> > > > > > I tried setting up the beacons using IA
> > > > > > these were the observations I found while setting using IA.
> > > > > > 1)Automatic map formation is difficult.
> > > > > > i)I used 3 stationary beacons and one hedgehog. The stationary beacon's position was kept on moving then it became stable. Then I freezed the map.
> > > > > > ii) I tried now with 4 stationary beacons for more coverage. The beacons kept on moving like in the previous case but after sometime it became stable and one of the stationary was sitting on other. But in the real setup all 4 beacons were placed at different corners of the room. So I do not understand why the map is not correct map automatically.
> > > > > > 2)When working 3 stationary beacons in IA the dashboard gives a warning "Incorrect Map. Please Freeze Again". Why does this error come?
> > > > > > 3)The above error came when ever I added a new hedgehog to the system and the map was freezed. When I add hedgehogs in NIA particularly when map is freezed , this error wouldn't come then why in IA?
> > > > > > So I need your input from your side whether it is possible to set up 10 hedgehogs using IA so that I can have a correct location with a good update rate?
> > > > > > If possible, can you please provide detailed instructions in setting up the system?
> > > > > >
> > > > > > **If possible can you help me arrange an online session with my team so that all doubts can be clarified in a single meeting as we need to deliver some things very soon.**
> > > > > >
> > > > > > Thanks & Regards,
> > > > > > Sriniketh Konduri.
> > > > > >
> > > > > > On Mon, Sep 28, 2020 at 5:36 AM Marvelmind Robotics (Vladimir) <vladimir@marvelmind.com> wrote:
> > > > > > > Hello,

https://outlook.office.com/mail/inbox/id/AAQkADRkZDY4YmQ0LWI4NWQtNGFhNC1hMzdlLTJlMTNhYTMxZTFlNAAQADdw0Tlq3VBBhzhaviMIC... 4/8

On your screenshots I see the update rate is less than 1 Hz, it looks too slow for IA even for 10 hedgehogs.
Or now you have returned to NIA?
Do you use the latest version of software (v6.240)?
Please share a file generated by the 'Save map' button in dashboard, we will check the settings and try to repeat and fix the issue with empty beacons_pos_a topic.

BR,
Vladimir

On Sat, Sep 26, 2020 at 11:40 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:

> HI,
> First thanks for the reply,
> Answer for your questions are below.
> We always recommend first to achieve basic tracking with 1 mobile beacon (hedgehog). Then the same with ROS or other advanced connectivity. And only then to move to more complex settings: more hedges, more submaps, etc.
> - Have you achieved the perfect tracking with one hedge?
> - With 3 hedges?
> I have achieved the tracking for one hedgehog both on dashboard and ROS.
> The below situation was before
> But for three hedgehogs sometimes the one of the hedgehog won't be visible in the dashboard at all.
> And when it is visible on the dashboard it keeps on jumping from one place to another.(NOTE: The line of sight is not obstructed for any of the beacon and hedgehog)
>
> Now as you suggested I connected one by one :
> Now 10 hedgehogs were able to connect into the network.
> All the hedge beacons were visible into the dashboard map and the update rate was very slow as I connected 10 hedgehogs
> Even in ROS I am able to get the data of all the beacons except for the stationary beacon data coordinates through **/beacons_pos_a.**
> The topic doesn't have any messages at all when I echo it.
> <Screenshot from 2020-09-26 15-08-24.png>
> I tried as your colleague suggested to run the subscriber test node. I am able to observe the hedgehog on the rviz but not the stationary beacons.
> As I see in the code, the code uses **/beacons_pos_a** topic to get the data of the stationary beacons to make the visualization in RIVZ. But the topic **/beacons_pos_a** itself does not have any data. So its obvious that it wont be visualized in rviz.
> **So why the topic /beacons_pos_a is empty?**
> <Screenshot from 2020-09-26 16-18-13.png>

- Which stationary beacons did you try to use? What ultrasonic frequencies have they?
- Make sure you are following the Placement Manual for correct placement of stationary beacons
- It may be easier for you to build 2D submap first with 2 stationary beacons. Again, the same logic: try simple configuration first and only then move to the more complex ones

marvelmind Super beacon outdoor has been used for stationary beacons. The frequency for each and every stationary beacon is different as suggested by the manual. (19,25,31,45KHz).
The IA setup is very important for the my project because the NIA update rate of hedeghog is very slow. So please guide me in setting up the beacon network using IA as well as ROS interface.
Thanks & Regards,
Sriniketh Konduri.
Thanks & Regards,
Sriniketh Konduri.

On Thu, Sep 24, 2020 at 11:46 PM Marvelmind Robotics <info@marvelmind.com> wrote:

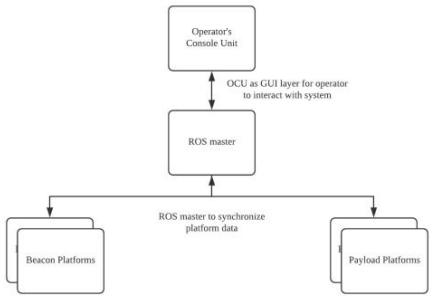
> Hello Sriniketh,
>
> Please, see my replies and comments below.
>
> Also, as usually, please, closely follow the Operating Manual and update the SW for beacons, modem and the Dashboard from the same SW pack. Always. Don't mix the SW from different SW releases.
>
> Kind regards,
> Maxim
>
> On Thu, Sep 24, 2020 at 8:35 AM sriniketh 22 <kondurisriniketh22@gmail.com> wrote:
>
>> Hi,
>>
>> Recently I bought 18 super outdoor beacons. The beacons are of different frequency. I used 4 beacons as stationary and three as mobile beacons. I established the beacon network through NIA.
> We always recommend first to achieve basic tracking with 1 mobile beacon (hedgehog). Then the same with ROS or other advanced connectivity. And only then to move to more complex settings: more hedges, more submaps, etc.

> - Have you achieved the perfect tracking with one hedge?
> - With 3 hedges?
>
>> Then I connected them over ROS as well. Now I have a few doubts which need to be addressed.
>> 1)When I connect over ROS, mobile beacon coordinate data can be accessible over ROS topics like hedge_pos_a, etc. the stationary beacon coordinates were unable to access over the ROS topic i.e beacons_pos_a. For my project this data is very essential to view it on RVIZ.
> My colleague in CC will be able to comment and advise. But I strongly recommend you to achieve proper tracking and share with us the result, before you jump to ROS. Please, do
>
>> 2)When I connected 2 mobile beacons the system worked well but when I connect 3 mobile beacons the system becomes unstable. One of the mobile beacons keeps on oscillating on the map, which creates a major problem for my robot autonomy as localization is unstable.
> - What happens, if you leave the two mobile beacons including the one that is still jumping? Is it the issue of more than 2 hedges or the issue of that particular hedge?
> - Try the suggested variant and let's see
> - Also try just one hedge. First, "good' one and then the one that is jumping. But all - one by one. Not two at once
>> 3)I wanted to connect all my beacons in an INVERSE ARCHITECTURE scheme. as I want to connect 10 mobile beacons so as to get the good update rate. In IA the stationary beacons cannot create the correct map automatically. The stationary beacons keep on oscillating and most of the time a beacon won't be seen on the map even when I awake them.
> - Which stationary beacons did you try to use? What ultrasonic frequencies have they?
> - Make sure you are following the Placement Manual for correct placement of stationary beacons
> - It may be easier for you to build 2D submap first with 2 stationary beacons. Again, the same logic: try simple configuration first and only then move to the more complex ones
>
>> 4) And in IA the mobile beacons were not visible even after awakening them.
> - Probably, some steps are not done correctly ... Not visible where? On the map? Or not awaken at all?
>
>> Please help me on this issue as soon as possible as many outdoor beacons have been bought.
>>
>> Thanks & Regards,
>> Sriniketh Konduri.

User Manual

User Manual

System Overview



Overall System architecture

The Raptor system consists of the following:

- Payload platform, the robot which carries the payload and executes formation orders
- Beacon platform, robot which carries stationary beacons for payload platform localization
- ROS master, main controller in which all information received, processed, and sent
- Operator's Console Unit, graphical user interface for the operator to command platforms.

This document describes the steps to set up the above system.

Required Machines

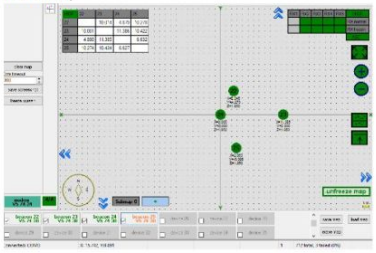
- ROS Master, an Ubuntu machine with ROS Melodic, ROSBridge Suite and ROSMaster Raptor package installed
- Operator Console Unit (OCU), a Windows machine with the OCU app installed
- Beacon and payload platforms with batteries plugged in
- Raptor Wifi Router with all static IPs saved

Setup Procedure

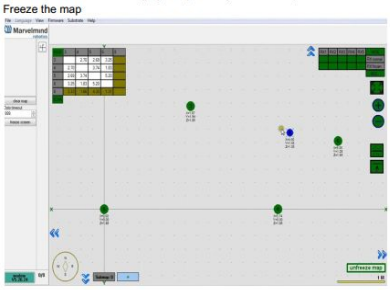
1. Turn on Raptor Wifi, connect ROS Master and OCU to it.
2. Turn on ROS Master and launch the following in separate terminals:

ROS module	Command
ROSBridge	roslaunch rosbridge_server rosbridge_websocket.launch
Map Server	roslaunch gps_navigation start_map_server.launch

3. Run OCU and input the ip address of ROS Master when prompted
4. *Turn on beacon platforms
5. *Teleop beacon platforms to different corners of operational area to form map using Manual Movement Mode in OCU
6. In Marvelmind Dashboard,
   a. Add the stationary beacons from the beacon platforms and position them
   b. freeze submap



   c. Add all the mobile beacons (payload platform) onto the map
   d. Freeze the map



7. Turn on the payload platforms
8. Wait for payload platforms to boot up. Payload platforms will move slightly to indicate system ready on ROS and OCU will indicate activated platforms as blue
9. System is now ready for commands from OCU

*In the production system, beacon platforms are not used as placement is too low for Marvelmind beacons to function. Instead, Marvelmind beacons are placed 2m above ground to maintain line of sight with all payload platforms.
** For more information on Marvelmind beacon please visit:
https://marvelmind.com/pics/marvelmind_navigation_system_manual.pdf