
Indoor Mapping

System Design
Specifications with
Component details

Group 3

Abdel-Latif, Sari	0840264
Batth, Chanderdeep	0856000
Bishara, Marc	0858892
Elsaftawy, Mahmoud	0951912
Mansour, Ahmed	0857403
Wahid, Fahim	0965325

24/01/2014

Table of Contents

List of figures	2
Revision Table	3
Introduction	4
Purpose	4
Scope	4
References	4
Note	4
System Overview	5
Module breakdown diagram	5
System Modules Description	6
System's Interfaces description	7
System Hardware overview	13
Definitions, Acronyms and abbreviations	14
System Monitored and Controlled Variables	15
System Constants	15
System functional specifications	16
Sensor Interface Module (SIM)	16
Carrier	21
Modular Mapping Unit	27
Planner	33
POI	45
Communication Module	54
Visualizer	58
Command Center Module	63
Normal Operation	66
Normal Use Case	66
Undesired Behavior	66
Risk mitigation measures	66
Division of Labor	66

List of figures

Figure 1 - System Modules breakdown showing interfaces, monitored and controlled variables	5
Figure 2 - Hardware overview	13
Figure 3 - SIM Module Functional Breakdown.....	17
Figure 4 - Breakdown of the MMU Software.....	31
Figure 5 - Planner Functional Breakdown	34
Figure 6 - Visualizer Functional Breakdown	60
Figure 7 - Command Center Module Functional Breakdown	64

Revision Table

Version	Date	Author	Description
0.0	11 Nov, 2013	M Bishara	Initial release
0.1	13 Nov, 2013	S.Abdel-Latif	Adding initial Command Center Interface Designs
0.2	22 Nov, 2013	M Bishara	Completed overall system description sections
0.3	25 Nov, 2013	M Bishara	Added initial MMU Design Specifications
0.4	25 Nov, 2013	F Wahid	Added Carrier Module Design Specifications
0.5	25 Nov, 2013	M ElSaftawy	Added Planner Module Design Specifications
0.6	25 Nov, 2013	S.Abdel-Latif	Adding Visualizer Module Design Specifications
0.7	25 Nov, 2013	ChanderdeepBatth	Added POI Module Design Specifications
0.8	25 Nov, 2013	S. Abdel-Latif	Adding module interface description for visualizer
0.9	30 Nov, 2013	M. Bishara	Added Module interfaces and SIM white box
0.9.1	02 Dec, 2013	F Wahid C Batth M Bishara M ElSaftawy	Updated Carrier, POI and Planner modules to version 0.9.1 of each respectively
0.9.2			
0.9.3			
0.9.4			
0.9.5	03 Dec, 2013	A Mansour	Updated MMU
0.9.6	04 Dec, 2013	M ElSaftawy	Editing and reading over the Doc
0.9.6.1	04 Dec, 2013	M Bishara	Updating the Communication Module
0.9.6.2	04 Dec, 2013	A Mansour	Finished making corrections to the inputs table for the MMU
0.9.7	04 Dec, 2013	S Abdel-Latif	Final Formatting and finalizing
1.0	23 Jan, 2014	M Bishara	First iteration of System Component Design
1.0.1	23 Jan, 2014	M Bishara	Modified and expanded on system component design for the SIM
1.0.2	23 Jan, 2014	M Elsaftawy	Expanded and modified system component design for the Planner
1.0.3	23 Jan, 2014	A Mansour	Added component design for the MMU
1.0.3.1	23 Jan, 2014	F Wahid	Component Design, MIS and MID added for Carrier
1.0.3.2	23 Jan, 2014	C Batth	Expanded and modified system component design for the POI
1.0.4	23 Jan, 2014	S Abdel-Latif	Added and Expanded on component design for the Communication Module, Visualizer and Command Center Module
1.0.5	24 Jan, 2014	S Abdel-Latif	Final Formatting and finalizing

Introduction

Purpose

This document will contain the detailed description of the indoor mapping system's implementation. It fully describes all the modules along with their interfaces, hardware design and software design. Reader of this document should be able to fully replicate the system implementation to meet the requirements as specified by the System Requirements Specifications V 0.24 and System Goals V 3.

Scope

The document does not include any of the top system's requirements. The document shall clearly state all the system requirements for the different modules of the system that have been formulated to satisfy the high level system requirements as specified in the system requirements specifications.

The Document assumes prior knowledge of the system requirements and system goals.

With regards to hardware, the document will include both requirements and details on the actual hardware to be used on the system.

With regards to software, the document contains a full module breakdown of the components in the system. Every module's software is documented with a white box showing the inputs and outputs and module breakdown components. Components are documented with prototypes and a theory of use. As well, any algorithms implemented have been documented with flow charts.

References

System Requirements specifications:

Group_3_Deliverable_2_Draft_System_Requirements_v0.24

Goals:

Group_3_Capstone Project Goals V3.pdf

Note

Variable Names are case insensitive. Variable names have been chosen to be as descriptive as possible however Upper and Lower case has no inherent meaning.

(i.e. M_Human_Ctrl is the same as m_human_ctrl)

System Overview

Module breakdown diagram

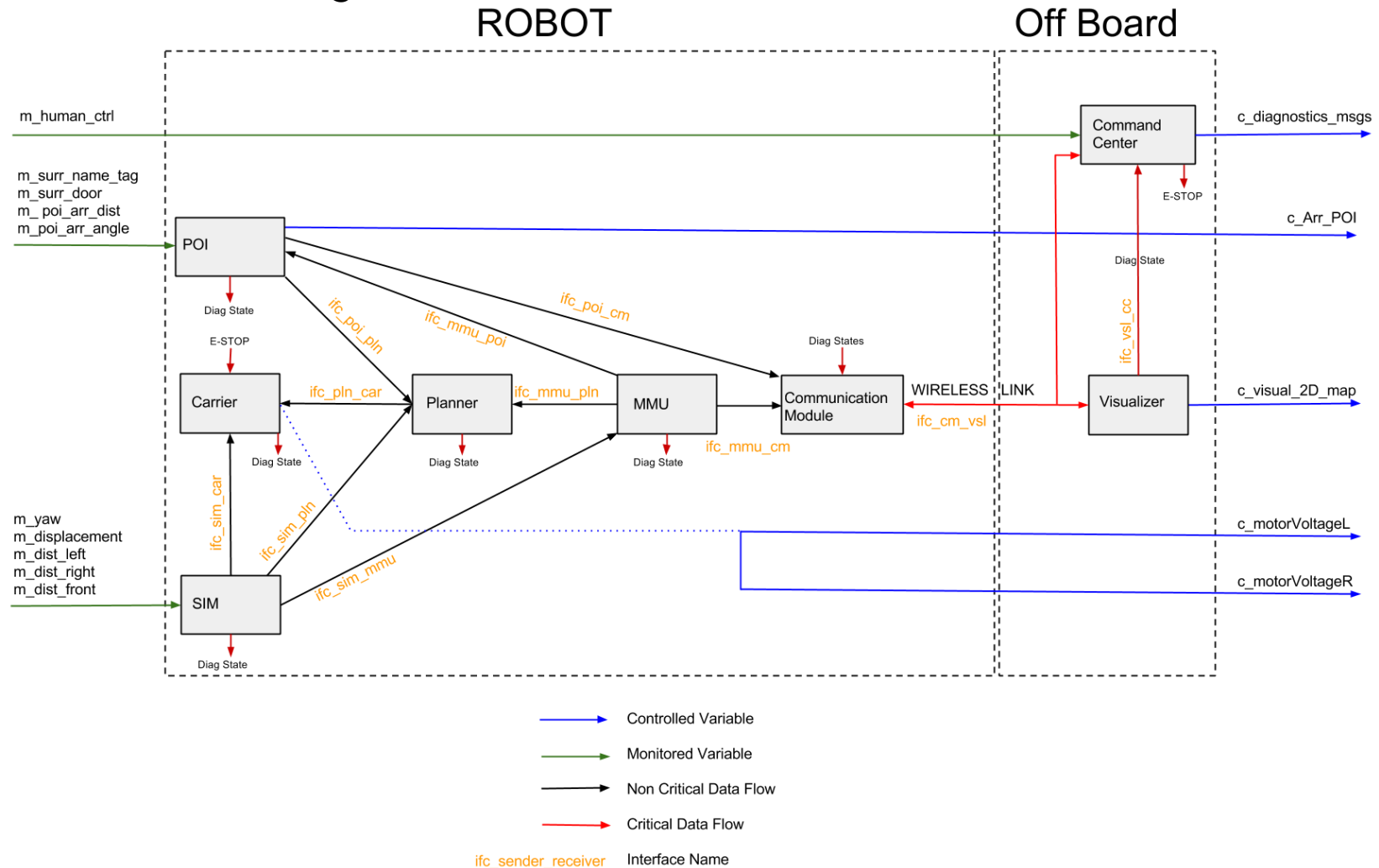


Figure 1 - System Modules breakdown showing interfaces, monitored and controlled variables

System Modules Description

Module	Purpose	PC
SIM	Sensor Interface Module connects to all the sensors. It uses drivers to abstracts all the hardware from other modules.	Arduino
Carrier	Carrier module carries all the hardware and translates planner commands into actual movement. It includes the control system and feedback for insuring proper movement	
MMU	Modular Mapping Unit uses outputs from the SIM to create a map of the environment and localize the unit on that map	Beagle Black
Planner	Planner plans the exploration and movement. It monitors all the sensors and sends its commands to the carrier for execution	
POI	Point of Interest module monitors doors and nametags in the environment to locate them on the map	
Communication Module	The purpose of the communication module is to keep connect the robot with the off-board system. It handles all the buffering, sending and receiving of data for all the modules	
Visualizer	The Visualizer is off-board, receiving data from the MMU and POI it constructs the visual map for the user	Laptop
Command Center	The Command Center houses the GUI interface to the human allowing them to monitor the status of the various modules, the last known location of the Bot and an option to execute an Emergency Stop.	

System's Interfaces description

This section explains the data flow lines shown in Figure 1, it explains the details of the interfaces and their design.

Messages are strings formatted as following:

SendingModuleName_ReceivingModuleName_TimeStamp_Message

lfc_poi_cm

- Data flow from POI Module to Communication Module
- Variables:

Variables	Type	Description
Current_POI	List of undefined size. Each element contains: POI Name, (String) POI Description, (String) Location X, (Int) Location Y (Int)	Every time a new POI is identified it is sent to the visualizer via the communication module ie. "Door", "", 15, 20 OR "Name Tag", "221", 15, 20

- Format:
 - Follows the universal message format
 - Asynchronous
- Hardware
 - TCP/IP Communication over an Ethernet Cable

lfc_mmu_poi

- Data flow from MMU Module to POI Module
- Variables:

Variables	Type	Description
Pose	String of the following format: "X-Y-Yaw"	The pose information will be sent to the POI module at K_Pose_Map_Update_Frequency

- Format:
 - Follows the universal message format
 - Synchronous at a rate of K_Pose_Map_Update_Frequency (Constant defined in the MMU Module Design)
- Hardware
 - TCP/IP Communication over an Ethernet Cable

lfc_poi_pln

- Data flow from POIModule to Planner Module
- Variables:

Variables	Type	Description
Planner_Command	String of the following format: "X-Y-Yaw"	The exact location and orientation that the POI Module wants to be in will be sent to the planner for implementation

- Format:
 - Follows the universal message format
 - Asynchronous
- Hardware
 - TCP/IP Communication over an Ethernet Cable

lfc_pln_car

- Data flow from PlannerModule to Carrier Module. Completed Message flows back from the Carrier to the Planner.
- Variables:

Variables	Type	Description
Desired_Speed	String enum. Values are "Stop", "Medium", "High"	The Planner asynchronously sends the speed to the carrier to proceed in.
Desired_Angle	Int between 0 and 360	The Planner asynchronously sends the angle to the carrier to proceed in relative to absolute, initial orientation
Completed	String	Sent back from the carrier to the planner to confirm that the requested angle, speed have been acheived

- Format:
 - Follows the universal message format
 - Asynchronous communication. The planner waits for a "Completed Message" from the Carrier before requesting new speed or angle.
- Hardware
 - Serial communication over USB. Planner is the computer and Carrier seen as a device.

lfc_mmu_pln

- Data flow from MMU Module to POI Planner
- Variables:

Variables	Type	Description
Pose	String of the following format: "X-Y-Yaw"	The pose information will be sent to the POI module at K_Pose_Map_Update_Frequency
2D_Map	Linked list each element indicating a detected wall and storing its location in x and y and string	Each time a wall is detected a new entry is added to the list with its x and y coordinates

- Format:
 - Follows the universal message format
 - Synchronous at a rate of K_Pose_Map_Update_Frequency defined in the MMU
- Hardware
 - Pose: TCP/IP Communication over loop back IP (127.127.0.1) since planner and mmu reside on the same computer
 - Map: Shared memory. The map is too large to be transferred on Ethernet as such it will be stored on memory and accessed by both the planner and MMU

lfc_mmu_cm

- Data flow from MMU Module to POI Module
- Variables:

Variables	Type	Description
Pose	String of the following format: "X_Y_Yaw"	The pose information will be sent to the POI module at K_Pose_Map_Update_Frequency
Map_Updates	Linked list each element indicating a detected wall and storing its location in x and y and string	The linked list contains all new updates to the map at K_Pose_Map_Update_Frequency

- Format:
 - Follows the universal message format
 - Synchronous at a rate of K_Pose_Map_Update_Frequency defined in the MMU design
- Hardware
 - TCP/IP Communication over loop back IP (127.127.0.1) since Communication Module and mmu reside on the same computer

lfc_sim_pln

- Data flow from Sensor InterfaceModule to Planner Module
- Variables:

Variables	Type	Description
dist_left	Real positive Integer	Describing the distance to the wall on the left if one has been detected
dist_right	Real positive Integer	Describing the distance to the wall on the right if one has been detected
dist_front	Real positive Integer	Describing the distance to the wall in front if one has been detected

- Format:
 - Follows the universal message format
 - Synchronous at a rate of k_sensor_update_frequency defined in the sensor interface module
- Hardware
 - Serial communication over USB. Planner is the computer and SIM seen as a device.

lfc_sim_mmu

- Data flow from Sensor InterfaceModule to Modular Mapping Unit Module
- Variables:

Variables	Type	Description
Yaw	Integer between 0 and 360	Integer describing the current orientation of the carrier relative to the orientation at point zero in time
Displacement	Tuple of integers (x,y)	X and Y define the displacement from initial position at point zero
dist_Sensors_Array	Array of Tuples each element contains the distance reading and sensor's angle.	Describing the distances measured by each of the sensors on the robot

- Format:
 - Follows the universal message format
 - Synchronous at a rate of k_sensor_update_frequency defined in the sensor interface module
- Hardware
 - Serial communication over USB. MMU is the computer and SIM seen as a device.

lfc_sim_car

- Data flow from Sensor InterfaceModule to the Carrier Module
- Variables:

Variables	Type	Description
Actual_Yaw	Integer between 0 and 360	Integer describing the current orientation of the carrier relative to the orientation at point zero in time
Actual_Speed	Real Integer	Integer value of the current speed of the vehicle used by the carrier unit at feedback for the drivers

- Format:
 - Values stored in global variables accessible by both SIM and Carrier modules
 - Synchronous at a rate of k_sensor_update_frequency defined in the sensor interface module
- Hardware
 - Program memory since both the SIM and carrier are running on the same computer.

lfc_cm_vsl

- Data flow from CommunicationModule to visualizer Module
- Variables:

Variables	Type	Description
Message	String	Any Message with the destination module set as visualizer

- Format:
 - Follows the universal message format
 - Asynchronous
- Hardware
 - TCP/IP Communication over Wi-Fi internet network

lfc_vsl_cc

- Data flow from visualizer Module to command center Module
- Variables:

Variables	Type	Description
Message	String	Any Message with the destination module set as CommandCenter

- Format:
 - Follows the universal message format
 - Asynchronous
- Hardware
 - TCP/IP Communication over an Ethernet Cable

lfc_cm_cc

- **Critical**Data flow from Communication Module to the Command Center
- Variables:

Variables	Type	Description
Module_Log	String	Any Log info with the destination module set as CommandCenter. A log file is expected to be coming in from each module.
e-Stop Command	String	Message to the Carrier module with e-Stop command

- Format:
 - Follows the universal message format
 - Asynchronous
- Hardware
 - TCP/IP Communication over an Ethernet Cable

System Hardware overview

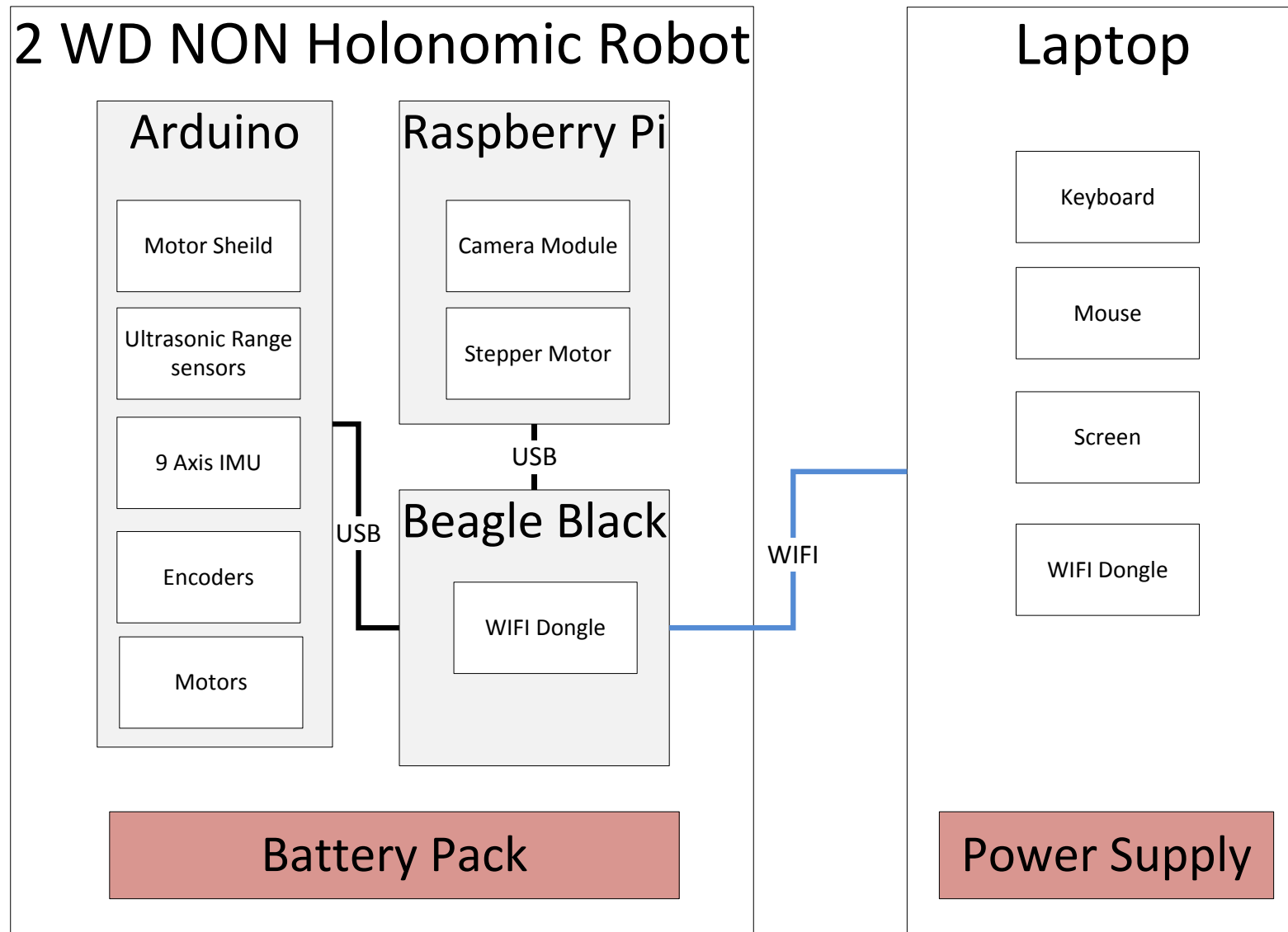


Figure 2 - Hardware overview

Definitions, Acronyms and abbreviations

MMU	Modular Mapping Unit
POI	Point of Interest Module
SIM	Sensor Interface Module
CAR	Carrier Module
PLN	Planner Module
CM	Communication Module
CC	Command Center Module
VSL	Visualizer Module

System Monitored and Controlled Variables

Variable	Units	Range		Accuracy
		Min	Max	
m_surr_name_tag	m ²	12 (Radius of 2 m around the bot)		0.5
m_surr_door	m ²	12 (Radius of 2 m around the bot)		0.5
m_poi_dist	m	0	2	0.2
m_poi_angle	m	0	180	5
m_yaw	Deg	0	360	2
m_displacement	m	All Real Numbers		5
m_dist_left	m	0	5	0.2
m_dist_right	m	0	5	0.2
m_dist_front	m	0	7.5	0.2
m_human_ctrl	NA	NA		100%
c_Arr_POI	NA	Real Positive Integers		80%
c_visual_2D_map	m ²	Real Positive Integers		5
c_left_motor_speed	m/s	0	0.3	0.01
c_right_motor_speed	m/s	0	0.3	0.01
c_diagnostics_msgs	NA	NA		100%

System Constants

Variable	Units	Range		Accuracy
		Min	Max	
K_system_Speed	m/s	0	0.16	0.02

System functional specifications

Sensor Interface Module (SIM)

Module Requirements Specifications

SIM_MR_01

The SIM shall abstract all sensor hardware from other modules and provide all their data at $k_Sensor_Update_Frequency$.

Rationale: To follow Software Best Practices, this module implements information hiding to abstract hardware from other modules

Fulfill requirement: SR_NFR_3-4

SIM_MR_02

SIM shall be able to monitor vehicle's actual displacement to within $k_displacement_error$

Rationale: This is the initial stage of pose estimation which will be enhanced in the MMU

Fulfill requirement: SR_FR_4-5

SIM_MR_03

SIM shall be able to monitor vehicle's actual speed to within k_speed_error

Rationale: This information will be used by the carrier module to insure the planner commands are executed to within their stated accuracy

Fulfill requirement: CAR_MR_02

SIM_MR_04

SIM shall be able to monitor vehicle's actual angle to within k_angle_error

Rationale: This information will be used by the carrier module to insure the planner commands are executed to within their stated accuracy

Fulfill requirement: CAR_MR_03

SIM_MR_05

The SIM shall report its status to the Diagnostic Centre module every k_log_period .

Rationale: A human shall have access to the status of the module

Fulfill requirement: CC_MR_03

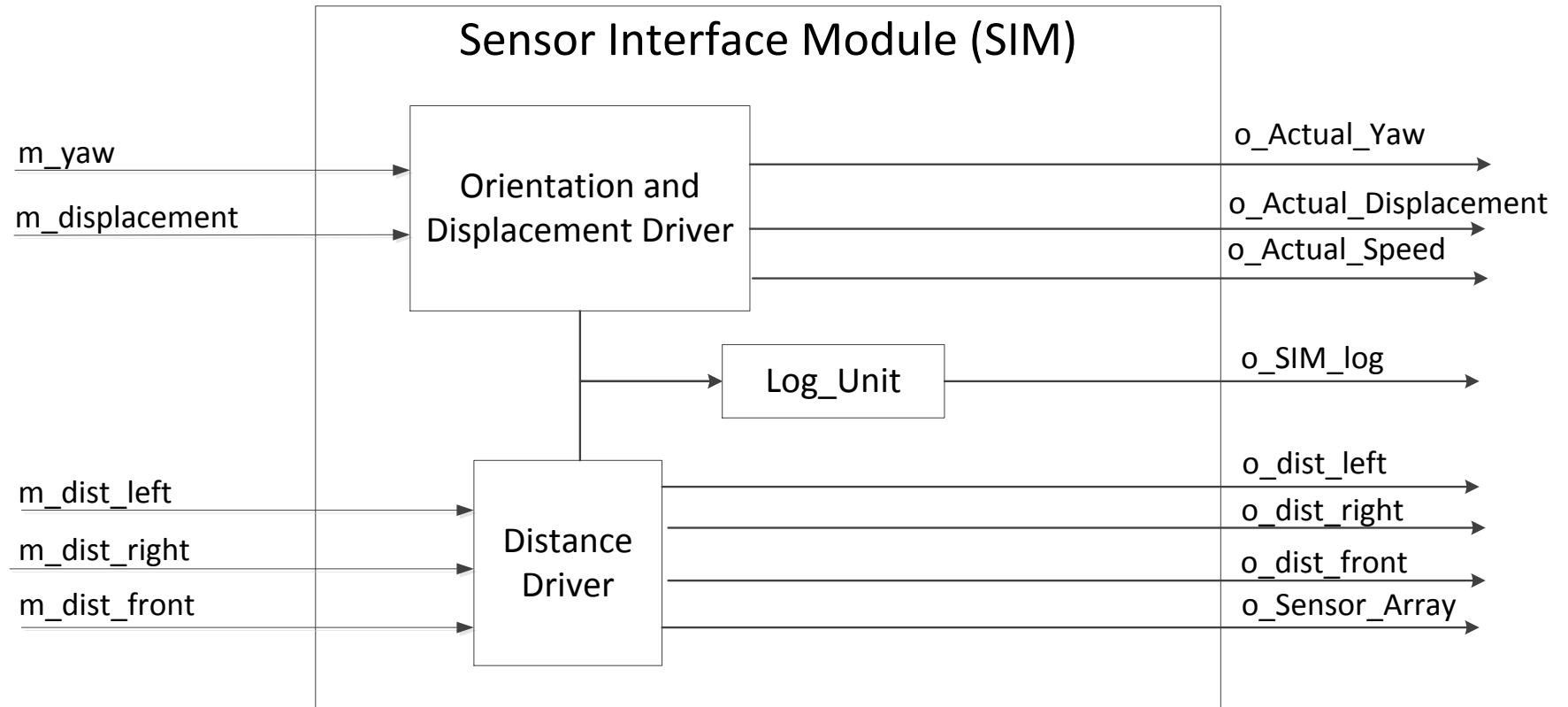
SIM Module Functional Breakdown

Figure 3 - SIM Module Functional Breakdown

Constants

Variable	Type	Unit	Description	Value
K_Sensor_Update_Frequency	Int	Hz	Frequency of reading and broadcasting sensor information	10
k_log_period	int	Hz	The rate at which the carrier sends its status to the Diagnostic Centre	10
k_speed_error	int	cm/s	maximum acceptable error of the speed	1
k_angle_error	int	degree	maximum acceptable error of the angle	1
k_displacement_error	Int	M	Maximum displacement estate error	50

SIM Inputs and Outputs

Variable	Units	Range		Accuracy
		Min	Max	
m_yaw	Deg	0	360	2
m_displacement	m	All Real Numbers		20
m_dist_left	m	0	5	0.02
m_dist_right	m	0	5	0.02
m_dist_front	m	0	7.5	0.02
o_Actual_Yaw	Deg	0	360	2
o_Actual_Displacement	m	All Real Numbers		50
o_Actual_Speed	m/s	All Real Numbers		0.01
o_SIM_Log	NA	NA		NA
o_dist_left	m	0	5	0.02
o_dist_right	m	0	5	0.02
o_dist_front	m	0	7.5	0.02
o_Sensor_Array	NA	NA		NA

Module Design Description (MIS & MID)

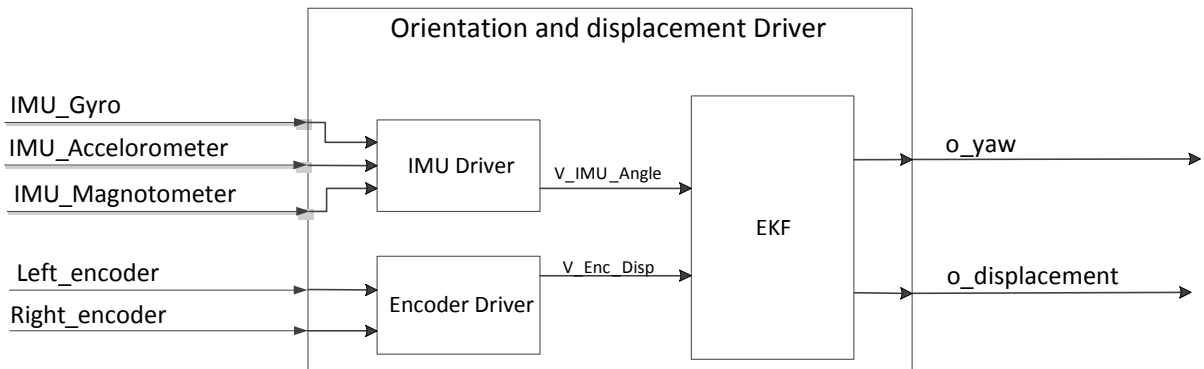
Orientation and Displacement driver

Orientation and displacement information are related to each other since they will be acquired as an output of an Extended Kalman Filter implementation that utilizes data from an IMU and encoders to calculate the displacement and orientation.

IMU data will contain an aggregation of a Gyroscope, Accelerometer and Magnetometer to estimate Angle and Displacement.

On Board DMP will be used to reduce noise and merge sensor data.

Two encoders will be used on to monitor wheel and provide second set of estimates on displacement and angle. Most of the IMU driver has been written by Jeff Rowberg on GitHub with an open license for fair use in projects. This code will be used and modified to fit our project. EKF filter has not been designed yet it will only be used if necessary.



IMU Driver functions breakdown

Function	Parameters	Returns	Description
dmpGetQuaternion	q,fifoBuffer	Void	Reads fifo buffer from sensor and fills quaternion array passed as pointer in function inputs
dmpGetGravity	gravity, q	Void	Uses Quaternion array to compute gravity and assigns value to the function input called gravity
dmpGetYawPitchRoll	ypr, q, gravity	Void	Uses gravity and the Quaternion array to calculate Yaw Pitch and Roll

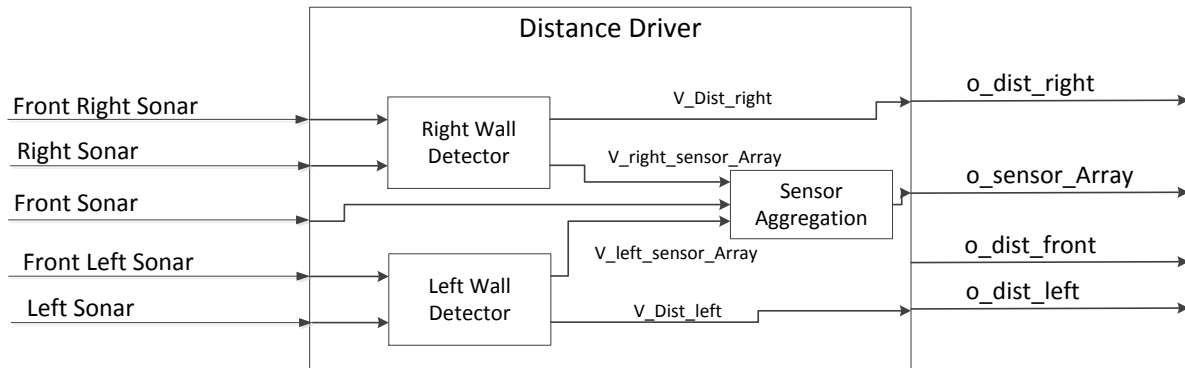
Encoder Driver functions breakdown

Function	Parameters	Returns	Description
RightPulseInterrupt	Void	Void	Triggers with every pulse from the right encoder. ISR includes ams timer from the last pulse to calculate speed. The function updates two global variables indicating the total travelled distance by the wheel and the current wheel speed.
LeftPulseInterrupt	Void	Void	Same as RightPulseInterrupt for left wheel

Distance Driver

Five ultrasonic sensors will be used to measure distances to wall in front and on both sides of the robot.

The data will be from the sensors will be aggregated to give better estimates for the planner. The driver will also output the sensor data raw for other modules to use at their discretion.



Right wall detector and left wall detector functions breakdown

Function	Parameters	Returns	Description
ReadSonar	Enum {Right, Left}	Int Distance	Reads PWM pin associated with sonar sensor to calculate distance to nearest object. Sensor outputs PWM with 147uS per inch. Function returns value in mm.

Sensor Aggregation functions breakdown

Function	Parameters	Returns	Description
AggregateSonarData	AggSonarArray, rawSonarArray	Void	Uses raw sonar data from five sonars and calculates three number (right, left and front distance). Some of the sonars are mounted on an angle so the function uses those angles and basic trigonometry to more accurately determine the wall distance

Carrier

Module requirements specifications

CAR_MR_01

The Carrier shall be able to turn without any displacement.

Rationale: There will be situations where the Planner plans to face at a different direction staying at the same position.

Fulfill Requirements: PLN_MR_02

CAR_MR_02

The Carrier shall be able to achieve the speed requested by the Planner within k_speed_error .

Rationale: The planner requires a minimum accuracy in order to calculate the correct decision.

Fulfill Requirements: PLN_MR_02

CAR_MR_03

The Carrier shall be able to achieve the orientation requested by the Planner within k_angle_error .

Rationale: The planner requires a minimum accuracy in order to calculate the correct decision.

Fulfill Requirements: PLN_MR_02

CAR_MR_04

If an obstacle is detected closer than $k_safe_distance$ in the direction of its movement, the Carrier shall ignore the Planner's request and move in the opposite direction until the distance is greater than $k_safe_distance$.

Rationale: This is a safety measure for connection loss with or incorrect decision by the planner.

Fulfill Requirements: SR_NFR_3-3

CAR_MR_05

If an emergency stop is requested by the Command Centre, the Carrier shall ignore the Planner's request and immediately stop moving.

Rationale: This ensures that a designated human operator has higher control over the carrier than the planner.

Fulfill Requirements: CC_MR_04

CAR_MR_06

The carrier shall report its status to the Diagnostic Centre module every k_log_period .

Rationale: A human shall have access to the status of the module.

Fulfill System Requirements: CC_MR_03

Constants

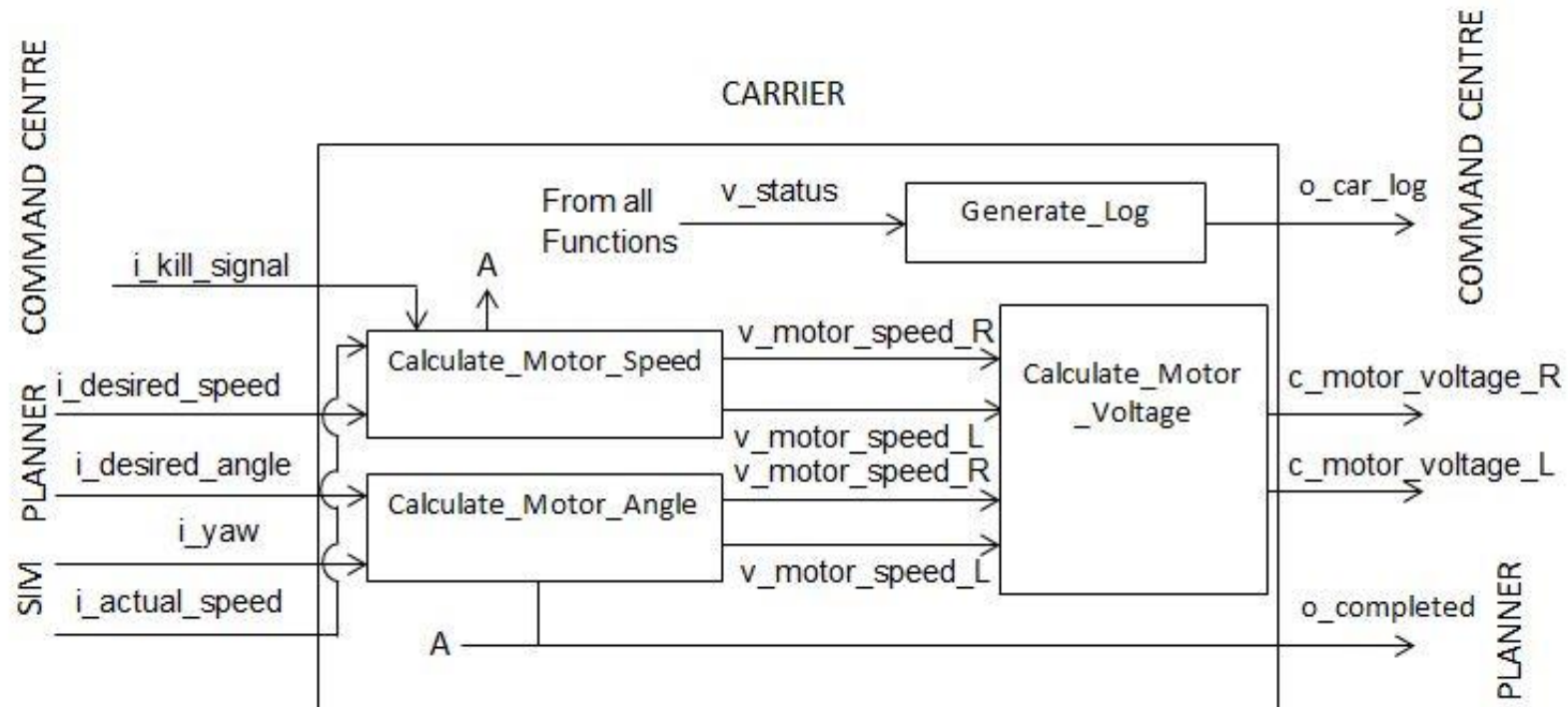
Variable	Type	Units	Description	Value
k_speed_error	int	cm/s	maximum acceptable error of the speed	1
k_angle_error	int	degree	maximum acceptable error of the angle	1
k_safe_distance	int	cm	A reasonable distance which must be maintained between carrier and obstacle	5
k_medium_speed	int	cm/s	Defined medium speed by Planner	12
k_high_speed	int	cm/s	Defined high speed by Planner	50
k_log_period	int	Hz	The rate at which the carrier sends its status to the Diagnostic Centre	10

Carrier Inputs and Outputs

Input/Output	Variable Names	Units	Description	Range
Input	i_desired_speed	cm/s	Desired qualitative speed from the Planner	i_desired_speed = enumSpeed.X See enum section for X
Input	i_desired_angle	rad	Desired angle of rotation from the Planner	$-\pi < \text{desiredAngle} < \pi$
Input	i_actual_speed	cm/s	Speed from Sensor	All real number
Input	i_yaw	pulse	Orientation from Sensor	$-\pi < \text{desiredAngle} < \pi$
Input	i_kill_signal	unitless	Emergency Stop Signal	i_kill_signal = {0, 1}
Output	c_motorVoltageL	rad/s	Left Motor Input Voltage	All real number
Output	c_motorVoltageR	rad/s	Right Motor Input Voltage	All real number
Output	o_carlog	-----	Status information about the carrier for diagnostic purpose	-----
Output	o_completed	-----	Status information about the carrier for diagnostic purpose	-----

Enum: enumSpeed

Enum Name (X)	Linear Velocity
speed_stop	0
speed_highF	$k_{\text{low_speed}} \pm k_{\text{speed_error}}$
speed_mediumF	$k_{\text{medium_speed}} \pm k_{\text{speed_error}}$

Module Functional Breakdown

Module Design Description (MIS & MID)

Module: Calculate_Motor_Speed

Calculates required motor speed to achieve the desired linear speed of the Carrier

Interface

Input

int i_desired_speed, i_actual_speed

Output

int v_motor_speed_R, v_motor_speed_L

Implementation

Theory of Operation

$v_motor_speed_R = f(i_desired_speed, i_actual_speed)$

$v_motor_speed_L = v_motor_speed_R$

Module: Calculate_Motor_Angle

Calculates required motor angle to achieve the desired orientation of the Carrier

Interface

Input

int i_desired_angle, i_yaw

Output

int v_motor_speed_R, v_motor_speed_L

Implementation

Theory of Operation

$v_motor_speed_R = f(i_desired_angle, i_yaw)$

$v_motor_speed_L = -v_motor_speed_R$

Module: Calculate_Motor_Voltage

Calculates required voltage to achieve the desired motor speed

Interface

Input

int v_motor_speed_R, v_motor_speed_L

Output

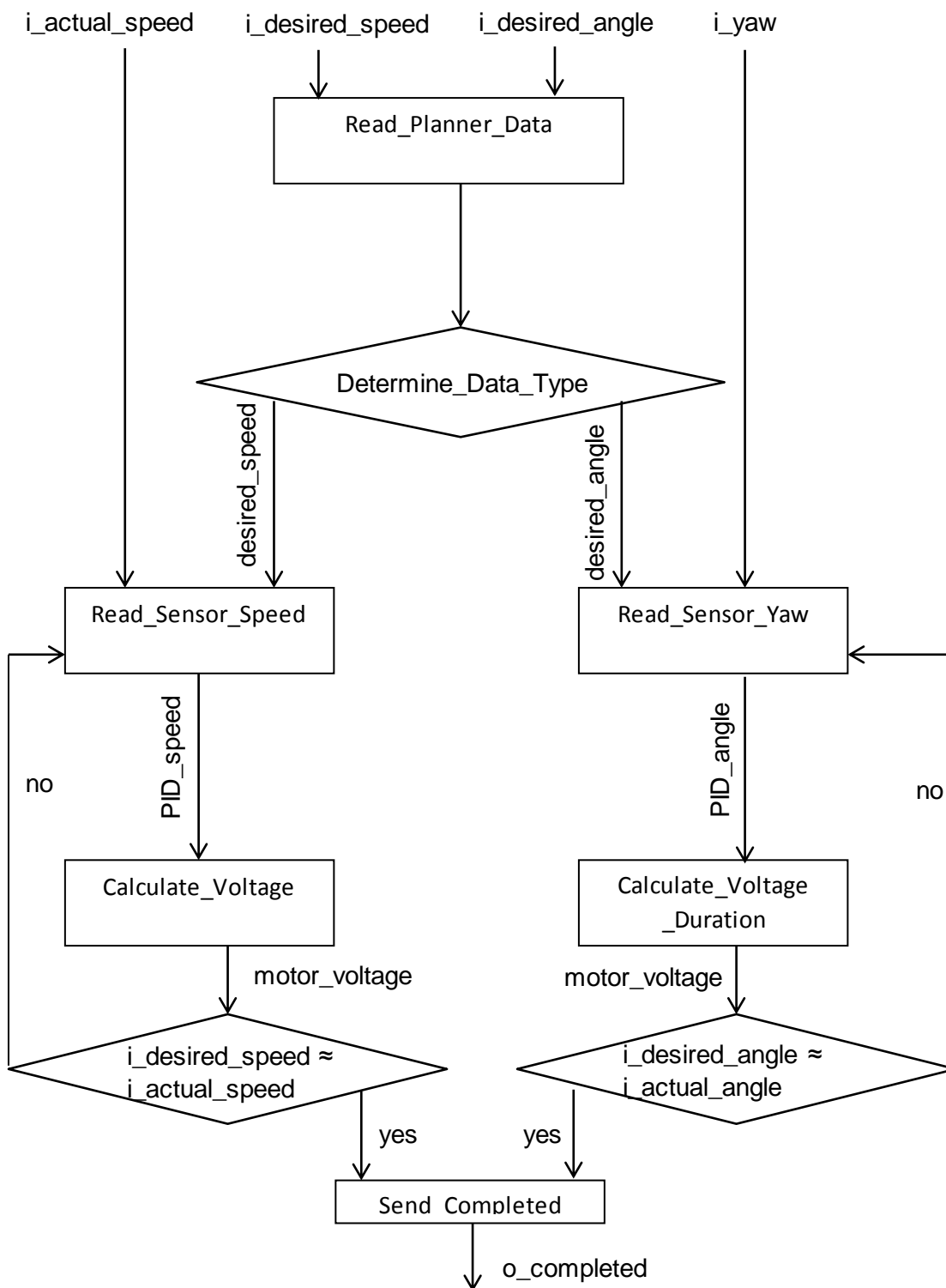
int c_motor_voltage_R, int c_motor_voltage_L

Implementation

Theory of Operation

$c_motor_voltage_R = f(v_motor_speed_R)$

$c_motor_voltage_L = f(v_motor_speed_L)$

Carrier Module Implementation Flowchart:

Modular Mapping Unit

Module requirements

MMU_MR_01

The MMU shall output a 2D map where continuous walls are represented by points placed no more than $k_{\text{minimum_map_density}}$

Rationale: This intermediary output is needed by the Planner and the Visualizer subsystems to function correctly

Fulfill Requirements:SR_FR_4-5

MMU_MR_02

The 2D map shall contain the pose of the MMU relative to the map accurate within $k_{\text{pose_tolerance}}$

Rationale: This intermediary output is needed by the Planner and the visualizer subsystems to function correctly

Fulfill Requirements:PLN_MR_01, and VSL_MR_04

MMU_MR_03

The 2D map shall be accurate within $k_{\text{map_tolerance}}$

Rationale: $k_{\text{map_tolerance}}$ was deemed reasonable based on G-01 which states that the target application is for the Information Technology Building. In this case an accuracy of $k_{\text{map_tolerance}}$ is sufficient to provide shoppers with location information

Fulfill Requirements:PLN_MR_01, and VSL_MR_04

MMU_SR_04

The MMU shall map without any prior knowledge

Rationale: The System Requirements document does not specify a specific map to be mapped, then the MMU should be map any building and must not rely on some prior knowledge to complete the task

Fulfill Goal: G-01

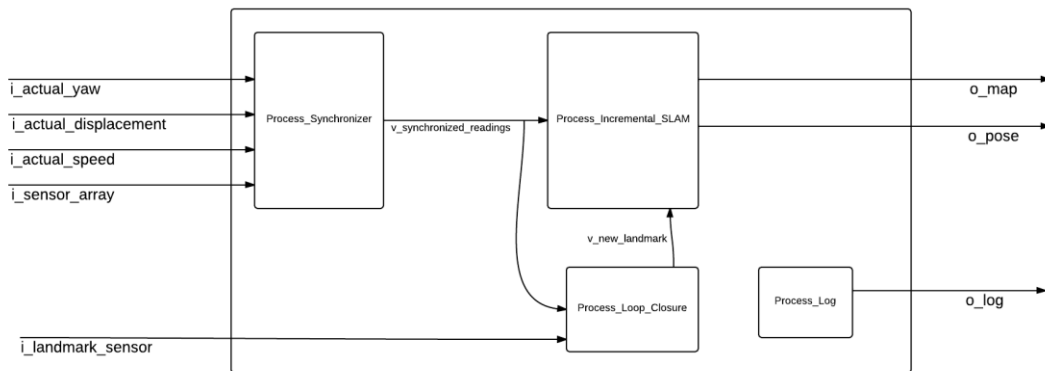
MMU_SR_05

The MMU shall function independently from the MMU motion

Rationale: G-02 states that mapping should be independent of the platform chosen for the mapping. This necessitates that the MMU be independent of the method of motion

Fulfills Goal: G-02

Module Functional Breakdown



Component design Considerations:

The MMU was chosen as a module because it hides the estimation of the pose from the other modules. The estimation of the pose is needed by a number of other modules. Hiding the details of pose and map creation allows other modules to develop based on the specified interfaces relying on a consistent interface between the modules.

Module Design Description (MIS & MID)

MIS

Please see interface MMU Module inputs and outputs

MID

Surroundings

Extracts information sent by the SIM and hides the hardware details of the SIM from the rest of the MMU

Pose Estimator

Uses the Surroundings and the Mapper and estimates the pose based on the surroundings

Mapper

Uses the Surroundings and the Pose Estimator to map.

Update

A scheduled module that send out and update to all the modules which use the MMU at **k_update_frequency**

Scheduling

Please see the Module requirements for scheduling.

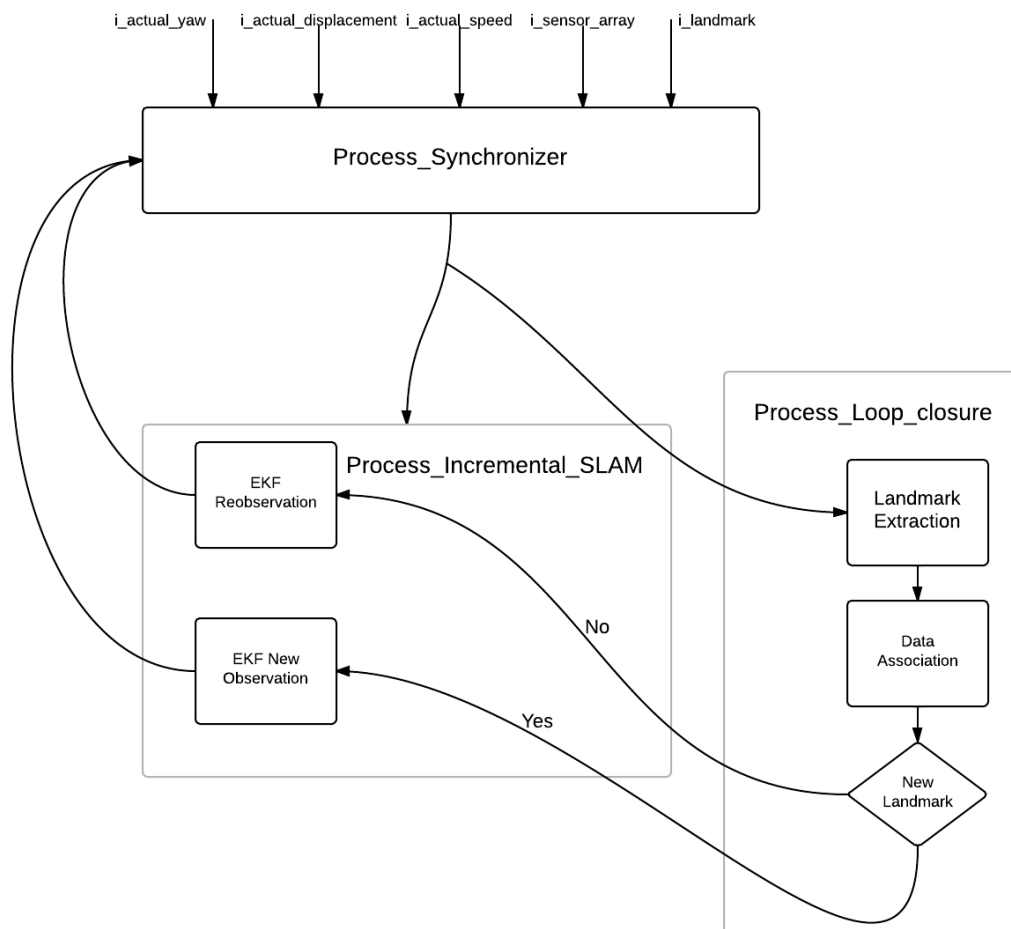
Internal Variables
v_synchronized_readings
v_new_landmark

Constants

Constant	Type	Units	Description	Value
k_minimum_map_density	unsigned int	samples/ meter	The minimum number of map samples per map	10
k_pose_tolerance	int	%	The tolerance range on the pose output of the MMU	$\pm 10\%$
k_map_tolerance	int	cm	The tolerance on the map output of the MMU	± 300
k_project_budget	int	CAD	The project budget in CAD	750
k_MMU_budget	unsigned int	CAD	The MMU budget in Canadian dollars	300

MMU Module inputs and outputs

Variable	Type	Input/ Output	Units	Description	Range	
					Min	Max
i_actual_yaw	R	Input	Deg	An estimate of the actual yaw from the IMU	[-180	180)
i_actual_displacement	R	Input	m,m	An estimate of the actual displacement	-5000, -5000	5000, 5000
i_actual_speed	+R	Input	m/s	An estimate of the actual speed	0	0.6
i_sensor_array	+R[]	Input	m	post EKF sonar readings	0	15
i_land_mark	N/A	Input	N/A	Recognized landmarks	N/A	
o_pose	R	output	m,m, Deg	A SLAM corrected pose	-5000, -5000, -180	5000, 5000, 180)
o_map	R	output	m,m	A stream of points at which there was a wall detected on the map	-5000, -5000	5000, 5000
o_MMU_log	char[]	output	N/A	Holds the status of the planner to be sent to the diagnostic center	N/A	

Slam process overview**Figure 4 - Breakdown of the MMU Software**

Algorithms

The algorithms used by MMU are shown in Figure 4 - Breakdown of the MMU Software

Odometry Design alternatives:

Option 1

Use IMU only, benefit is that the MMU will be completely independent of the type of carrier

Option 2

Use IMU and leave space for potential robot carrier Odometry information. Makes the assumption that there is a good chance the MMU is mounted on a robot of some sort that has its own Odometry so using its feedback can make the Odometry calculation more accurate.

Hardware:

IMU + Encoder

Cons:

We don't want to be dependent on the carrier at all so should we avoid it all together?

Loop Closure Design alternatives:

Option 1: Corners and gaps as landmarks

Use corners and gaps as landmarks, use sonar scanning to create a 2D image of the surroundings then RANSAC algorithm to fit lines for walls and look for spikes and 90 Degree angles for landmarks.

Pros: Sonar is easy to use, \$50 for 10 m range 5 cm accuracy 10+ Hz read rate

Cons: you could travel for up to 50m with no landmarks

Option 2: use doors for landmarks

Doors are big blobs that are easily identifiable with simple vision algorithm. Potentially use the Kinect sensor because it has depth calculation built into it.

Pros:

Very frequently occurring

Cons:

Some of them are too close to each other so not all doors can be used as landmarks

Kinect range is only 4 m

Image processing is more prone to errors

Planner

Module requirements specifications

PLN_MR_01

The planner shall insure that the complete floor has been mapped with no areas left unexplored.

Rationale: The planner drives the carrier; as such it shall insure that the carrier carries the MMU to all the areas of the building to ensure a complete 2D map of the floor.

Fulfill System Requirements: SR_FR_4-4

PLN_MR_02

The planner shall determine the immediate decision on the direction and speed in which the Carrier should proceed.

Rationale: The planner should output a velocity vector that is used to drive the carrier in the desired orientation and speed.

Fulfill System Requirements: SR_FR_4-2

PLN_MR_03

The planner shall maintain an absolute minimum distance of $k_{min_dist_to_wall}$ between the carrier and the left and right walls.

Rationale: The planner will ensure the different components of the system are safe by avoiding collision with the walls during the mapping process.

Fulfill System Requirements: SR_NFR_3-3

PLN_MR_04

The planner shall maintain a minimum distance of $k_{min_dist_POI}$ between the carrier and the right wall where the walls can be mapped.

Rationale: As required by POI, the Planner will ensure the POI is at an operational distance to the walls.

Fulfill System Requirements: SR_FR_4-3

PLN_MR_05

The planner shall report its status to the Command Centre module every k_{log_period} .

Rationale: As required by POI, the Planner will ensure the POI is at an operational distance to the walls.

Fulfill System Requirements: SR_NFR_3-3

Module Functional Breakdown

The following diagram - Figure 5 - is a high level description of the interactions of the different internal modules of the Planner unit. All these components will report to the Diagnostic Center.

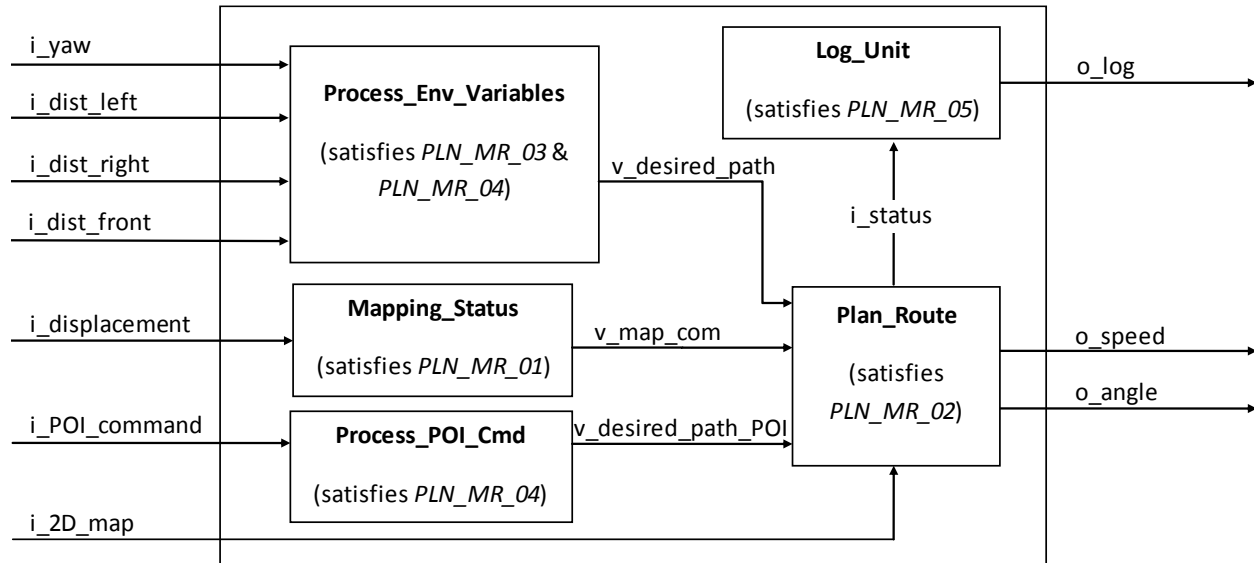


Figure 5 - Planner Functional Breakdown

Internal Variables
v_desired_path
v_desired_POI
v_status
v_map_comp (bool)

Constants

Variable	Type	Units	Description	Value
k_min_dist_to_wall	long	cm	The minimum distance between the system and the left, right, and front walls	100
k_min_dist_POI	long	cm	The POI needs to be at a minimum distance to the walls to function properly	150
k_planner_ref_rate	int	Hz	The rate at which the Planner receives and processes the four environment variables from the Sensor module	100
k_log_period	int	Hz	The rate at which the Planner sends its status to the Command Centre	10

Planner Inputs and Outputs

Variable	Type	Input/Output	Units	Description	Range	
					Min	Max
i_2D_map	N/A	Input	N/A	A blueprint like of the target floor that shows where all the mapped walls are	N/A	N/A
i_yaw	+R	Input	degrees	The difference in angle between the system's current orientation and its orientation at time zero	0	360
i_dist_left	+R	Input	cm	The physical distance between the mapping unit and the wall to the left of it.	0	2000
i_dist_right	+R	Input	cm	The physical distance between the mapping unit and the wall to the right of it.	0	2000
i_dist_front	+R	Input	cm	The physical distance between the mapping unit and the wall to the front of it.	0	2000

i_displacement	+R	Input	cm	Holds the position of the mapping unit relative to the starting point. Is a vector that hold the distance and direction	0	10000
i_POI_command	ENUM	Input	N/A	Holds the action desired and necessary for the POI to operate.	ENUM_POI	
o_speed	cm/s	Output	N/A	Desired speed	ENUM_SPEED	
o_angle	+R	Output	Degrees	The desired angle with reference to the current orientation of the carrier	-180	180
o_pln_log	char[]	Output	N/A	Holds the status of the planner to be sent to the diagnostic center	N/A	N/A

Enum

Variable	Value
ENUM_SPEED	STOP = 0 cm/s MEDIUM = 16 HIGH = 50
ENUM_POI	STOP = ENUM_SPEED.STOP CONTINUE = ENUM_SPEED.MEDIUM REVERSE = - ENUM_SPEED.MEDIUM

Module Design Description (MIS & MID)

The following is a brief explanation of the main internal components that make up the Planner module:

Process_Env_Variables

The purpose of the Planner is to drive the Carrier in the hallway. Thus it is necessary for the Planner to be aware of the surrounding environment so it can accurately guide the Carrier. This is done by obtaining the four input variables from the Sensor module.

These variables are used to determine a new orientation, position and/or path for the Carrier to follow. These variables are critical to the safety and accuracy of the mapping system since they will be used to avoid hitting walls or obstacles. Once a new path is produced, it is passed to the *Plan_Route* unit to be processed further.

Function	Parameters	Returns	Description
init_planner	None	void	All global variables are initialized in this function only once at start-up. This includes variables keeping track of the Carrier's velocity, displacement, and distance to walls.
getSensors	char direc	int - distance to wall	Three sensors mounted on the front, right, and left of the Carrier. This function takes a character 'F', 'R', or 'L' and returns the sensor reading at the specified direction. The sensor reading is the distance between the Carrier and the walls around it.
main_planner	None	void	This function contains an infinite while loop which controls the flow of the Planner software. When integrated with the rest of the modules, this function must be called in the system's main loop and the while loop is then removed.

Process_POI_Cmd

In the case that the POI needs to go back to a certain area in the hallway, it is allowed to instruct the Planner to adjust its course momentarily. The input variable `i_POI_command` will hold the action desired by the POI which is then processed to produce the internal variable `v_desired_path_POI`. When instructed by the POI, the Planner will return to its original course to continue mapping. However, the POI will not be allowed to control the Planner while the Carrier is turning to avoid possible confusion with the route planning. For that reason, POI route-change requests are given higher priority than the main route.

Function	Parameters	Returns	Description
process_POI	enum <code>i_POI_command</code>	struct <code>v_desired_path_POI</code>	This function will be called by the POI module to request a change in direction. Refer to <i>Enum table</i> , page 34 for the definition of <code>i_POI_command</code> and a list of possible inputs. The function returns a struct containing the speed, direction, and angle of the new path.

Plan_Route

The unit assesses the current state of the Planner and uses the variables: `i_2D_map`, `v_desired_path` and `v_desired_path_POI` to decide on an appropriate path, orientation, or position.

Function	Parameters	Returns	Description
Detect_Wall_Turn_Algo	None	void	This function deploys the Right Hand Rule Algorithm which will guide the Carrier around hallways. This is done by sensing walls around the Carrier and always following the wall on its right side.
move_car	int8_t angle Speed_Level speed Int8_t dist	void	In this function, a struct is filled with the input parameters and sent to the Carrier module

Log_Unit

The current state of the Planner as well as any changes in its behaviour will be reported to the Diagnostic Center. The Diagnostic Center will have no control over the Planner.

Planner Module Implementation Flowchart

The flowchart below clarifies the order which the planner operates. References to different algorithms are included in the diagram to show how it satisfies its requirements specifications.

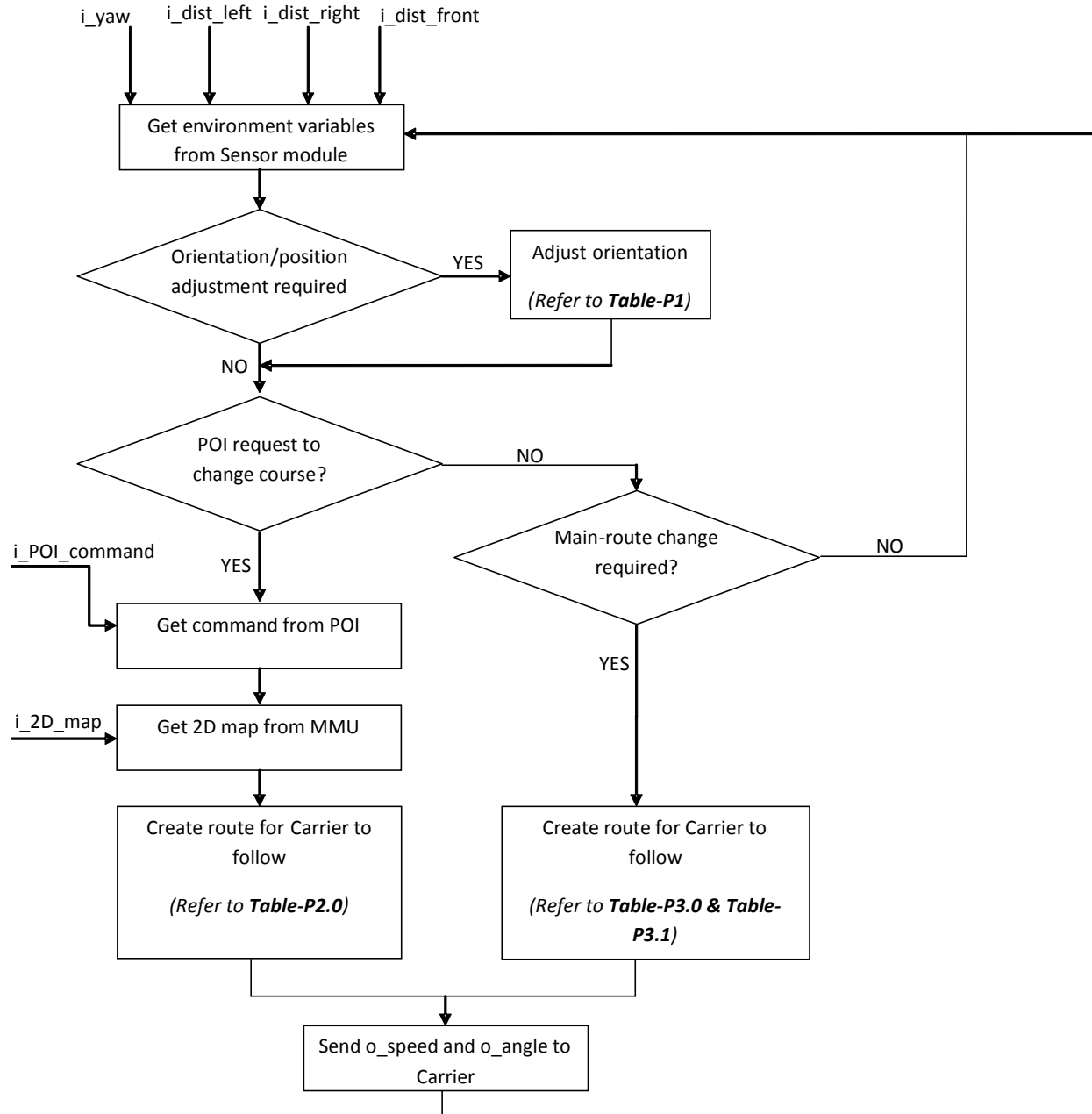


Figure P1 - Breakdown of the Planner Software

Algorithms

Table-P1.0 describes how the Planner decides to adjust the orientation of the Carrier. Req-P4 and Req-P5 are met using this algorithm which will ensure the Carrier is at a safe distance from the wall as well as a safe distance for the POI to function properly. Cases below are ordered in terms of higher priority first.

Case	Condition	Action
1	(i_dist_right < k_min_dist_to_wall) (i_dist_right < k_min_dist_POI) (i_dist_left > i_dist_right)	Shift Left o_angle = 15 o_speed = SPEED.MEDIUM
2	(i_dist_left < k_min_dist_to_wall) (i_dist_left < i_dist_right)	Shift Right o_angle = -15 o_speed = SPEED.MEDIUM
3	(i_dist_left == i_dist_right)	Do Not Shift o_angle = 0 o_speed = SPEED.MEDIUM

Table P1.0 - Adjust Orientation/Position Algorithm

Table-P2.0 describes how the Planner decides to modify its route according to the POI commands. The POI will be able instruct the Planner to stop or reverse the Carrier. Once the POI finishes controlling the Planner, it sends the command ENUM_POI.CONTINUE to continue on with the original route. Note that the Adjust Orientation Algorithm (**Table-P1.0**) will be disabled during this process.

Case	Condition*	Action
1	i_POI_command == ENUM_POI.STOP	Stop Carrier o_angle = 0 o_speed = ENUM_SPEED.ZERO
2	i_POI_command == ENUM_POI.CONTINUE	Continue Regular Route o_angle = 0 o_speed = ENUM_SPEED.MEDIUM
3	i_POI_command == ENUM_POI.REVERSE	Go Back o_angle = 0 o_speed = - ENUM_SPEED.MEDIUM

Table P2.0 - POI Override Algorithm

*All conditions assume the Carrier is not turning.

Table-P3.0 explains in detail how the Planner reacts to different changes in the environment (ie. walls). The Right Hand Rule algorithm ensures the entire hallway is mapped (Req-P1). However, given that a floor may have islands, following the Right Hand Rule will not suffice. Thus the algorithm must be modified so that it keeps track of these islands and go back to map them.

Case	Condition			Action
1	Wall right	No wall front	Wall left	Move Forward
2			No wall left	Move forward subhall_count++ subhall[Subhall_count].posX = disp_x subhall[Subhall_count].posY = disp_y subhall[Subhall_count].yaw=current_yaw +90
3		wall front	Wall left	Turn Left current_yaw = current_yaw + 90 o_angle = 90 o_speed = SPEED.MEDIUM
4			No wall left	
5	No wall right	Wall front	No wall left	Turn Right subhall_count++ subhall[Subhall_count].posX = disp_x subhall[Subhall_count].posY = disp_y subhall[Subhall_count].yaw=current_yaw+90 current_yaw = current_yaw -90 o_angle = -90 o_speed = SPEED.MEDIUM
6			Wall left	Turn Right current_yaw = current_yaw -90 o_angle = -90 o_speed = SPEED.MEDIUM
7		No wall front	No wall left	Turn Right //subhallway to the left subhall_count++ subhall[Subhall_count].posX = disp_x subhall[Subhall_count].posY = disp_y subhall[Subhall_count].yaw=current_yaw+90 //subhallway ahead subhall_count++ subhall[Subhall_count].posX = disp_x subhall[Subhall_count].posY = disp_y subhall[Subhall_count].yaw = current_yaw

Table P3.0: Right Hand Rule Algorithm

				<pre>current_yaw = current_yaw -90 o_angle = -90 o_speed = SPEED.MEDIUM</pre>
8			Wall left	Turn Right //subhallway ahead subhall_count++ subhall[Subhall_count].posX = disp_x subhall[Subhall_count].posY = disp_y subhall[Subhall_count].yaw = current_yaw current_yaw = current_yaw -90 o_angle = -90 o_speed = SPEED.MEDIUM

Table-P3.1 visually illustrates the different wall configurations that the Planner/Carrier will be facing. This should make Table-P3.0 easier to read.

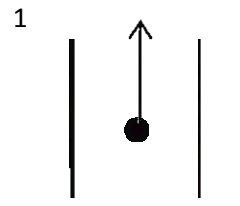
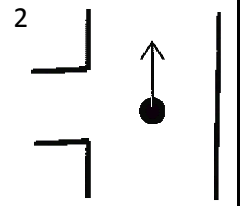
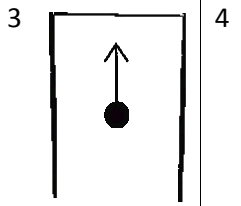
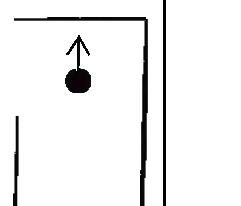
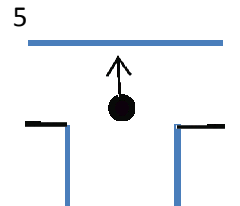
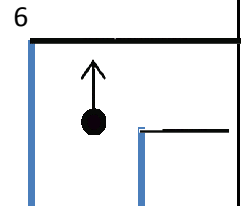
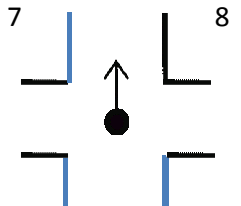
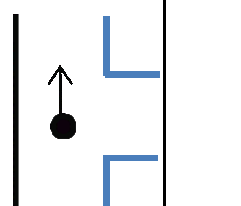
	No wall front		Wall front	
	wall left	no wall left	Wall left	No wall left
Wall Right	1 	2 	3 	4 
No wall Right	5 	6 	7 	8 

Table P3.1 Right Hand Rule Algorithm Continued

POI

Module Requirement Specification

POI_MR_01

The POI shall capture no less than 80% of the doors on the building floor.

Rationale: Specified as a goal and is therefore required for the system to succeed.

Fulfill System Requirements: SR_FR_4-3

POI_MR_02

The POI shall only capture important tags and room numbers during the mapping procedure.

Rationale: The system should not capture useless sticker tags or poster content. Only room number and type of room is required for this system.

Fulfill System Requirements: SR_FR_4-3

POI_MR_03

The POI module shall maintain a minimum distance of $k_{min_dist_POI}$ away from right wall

Rationale: Due to hardware limitation, the camera should be at least $k_{min_dist_POI}$ away in order to successfully capture all the required content.

Fulfill System Requirements:SR_FR_4-3

POI_MR_04

The POI shall be able to detect when a door is in the field of view and take an image.

Rationale: Taking images frequently would result in large CPU usage and use large amounts of memory.

Fulfill System Requirements:

POI_MR_05

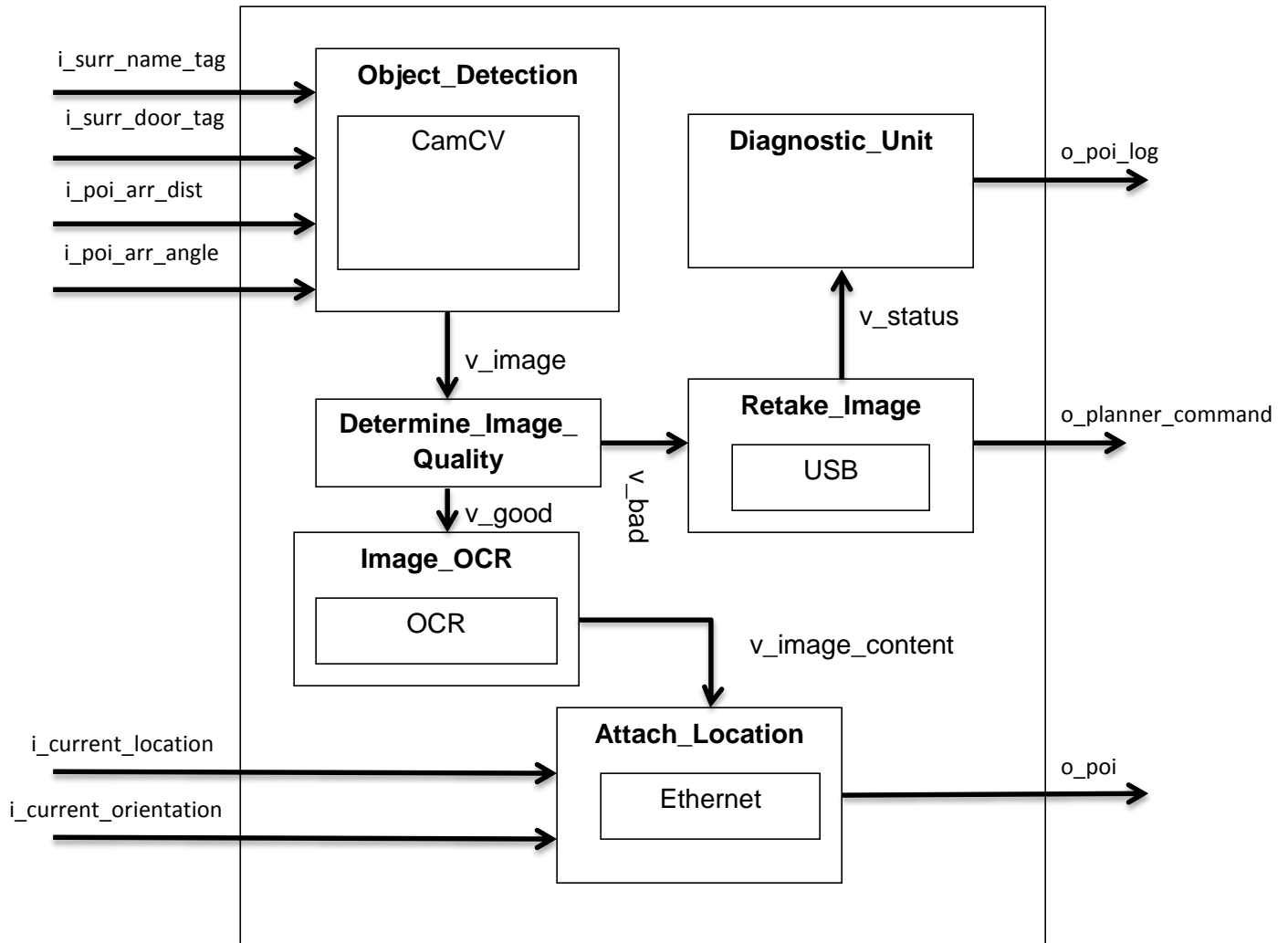
The POI shall communicate with planner to take carrier to POI location if image content is blurry.

Rationale: To ensure that POI_MR_01 requirement is met, the system needs to have good quality images

Fulfill System Requirements:SR_FR_4-3

Module Functional Breakdown

The following diagram is a detailed description of the POI module and the interfaces between different internal modules. The diagram shows a class breakdown and illustrates the responsibility of each class in the module. All the classes are defined in visible boxes and the whole module is part Main class. All the components will report to the Command Center.



Internal Variables
v_image
v_bad
v_good
v_status
v_image_content

POI Inputs and Outputs

Input/Output	Variable Names	Units	Description	Range
Input	i_surr_name_tag	N/A	The tags associated with each door captured	N/A
Input	i_surr_door_tag	N/A	The door numbers associated with each door	N/A
Input	i_surr_arr_dist	cm	The distance from the wall	k_min_dis_POI – 2000
Input	i_surr_arr_angle	rad	The angle of the door to POI module	0 – k_max_AOV_POI
Input	i_current_location	Cm	Coordinates of the current location of the carrier	All Real Numbers
Input	i_current_orientation	Rad	Current orientation of the carrier	0 – π
Output	o_planner_command	cm	Holds location information for the carrier to go back to	All Real Numbers
Output	o_POI	String[]	Location of point of interest along with the associated tags	-----
Output	o_POI_log	string	Status information about POI module for diagnostic purpose	-----

Constants

Variable	Type	Units	Description	Value
k_min_dis_POI	long	cm	The POI needs to be at a minimum distance to the walls to function properly	150
k_max_AOV_PIO	float	rad	The max angle of view the POI is limited to	0.94

Module Design Description (MIS & MID)

The following is a brief explanation of the main internal components of the POI module:

Object_Detection

This module will be responsible for detecting objects. Specifically, this module will detect when there is a door within the field of view of the camera. Once the door is detected, the camera will take an image of the door and the contents. The component will then feed the output to the Determine_Image_Quality component.

Class: CamCV			
Function Name	Parameters	Return Type	Description
RunVideoStream	N/A	Mat (Image container)	This function starts a video stream and returns true if stream starts successfully
HSVStream	videoStream	Mat	This function converts the camera stream to HSV space
BinaryImage	H_MAX, H_MIN, S_MAX, S_MIN, V_MAX, V_MIN	Mat	This function converts the video stream to a binary image by using HSV min and max threshold values
FindContour	binaryImage	Vector	This function is responsible for finding the contour of a binary image. This will result in a image matrix that will contain outlining of the binary image.

DrawContour	contour, hierarchy	N/A	This function is responsible for drawing the contour image. Input parameters contours and hierarchy are vectors that contain the contour and hierarchy values.
ImageMoment	contour	Float	This function uses the contour values of the image to find the moment of an image. This will allow the use to find the area of an image

Image_Quality_Check

Class: OCR			
Function	Parameters	Return Type	Description
Run	Image	String	Runs the OCR on the image and return the extracted text as a string

This module will determine if the image taken is not blurry. This could be achieved by computing the FFT and analyzing the results. The Fourier transform will show which frequencies are in the image. This module will send one of two outputs depending on the quality of the image. If the image is not of a good quality then it will output v_bad and this will initiate the Retake_Image to send command to Planner via USB connection.

Retake_Image

Class: USB			
Function	Parameters	Return Type	Description
Connect	USB Port	Boolean	This function is responsible for initializing the connection to the USB port.
Send	Data	Boolean	This function is responsible for pushing data to the USB port

This module is initiated if the signal is received from the Image_Quality_Check module. This will then create an instance of a USB connection and will push data to the USB

port, which is then received by the Planner module. If there is no need to take an image, then this component is never initiated.

Attach_Location

This module is initiated if the Image_Quality_Check component determines that the image quality is good. This output will initiate this component, which is responsible for attaching current locations received from the MMU module to the current point of interest. MMU module interfaces with POI module via Ethernet cable. The following table describes the Ethernet class used for interfacing:

Class: Ethernet			
Function	Parameters	Return Type	Description
Connect	TCP socket, port	Boolean	This function is responsible for initializing the connection to the Ethernet port.
Read	N/A	Boolean	This function is responsible for reading the data from the Ethernet port

Image_OCR

This module will be responsible for retracting important information from the image. Only room number and door tag will be required for this system. Any other posters or stickers will be removed from the image during the object detection phase. A simple OCR engine could be used to achieve this result. The following table gives an overview of the class

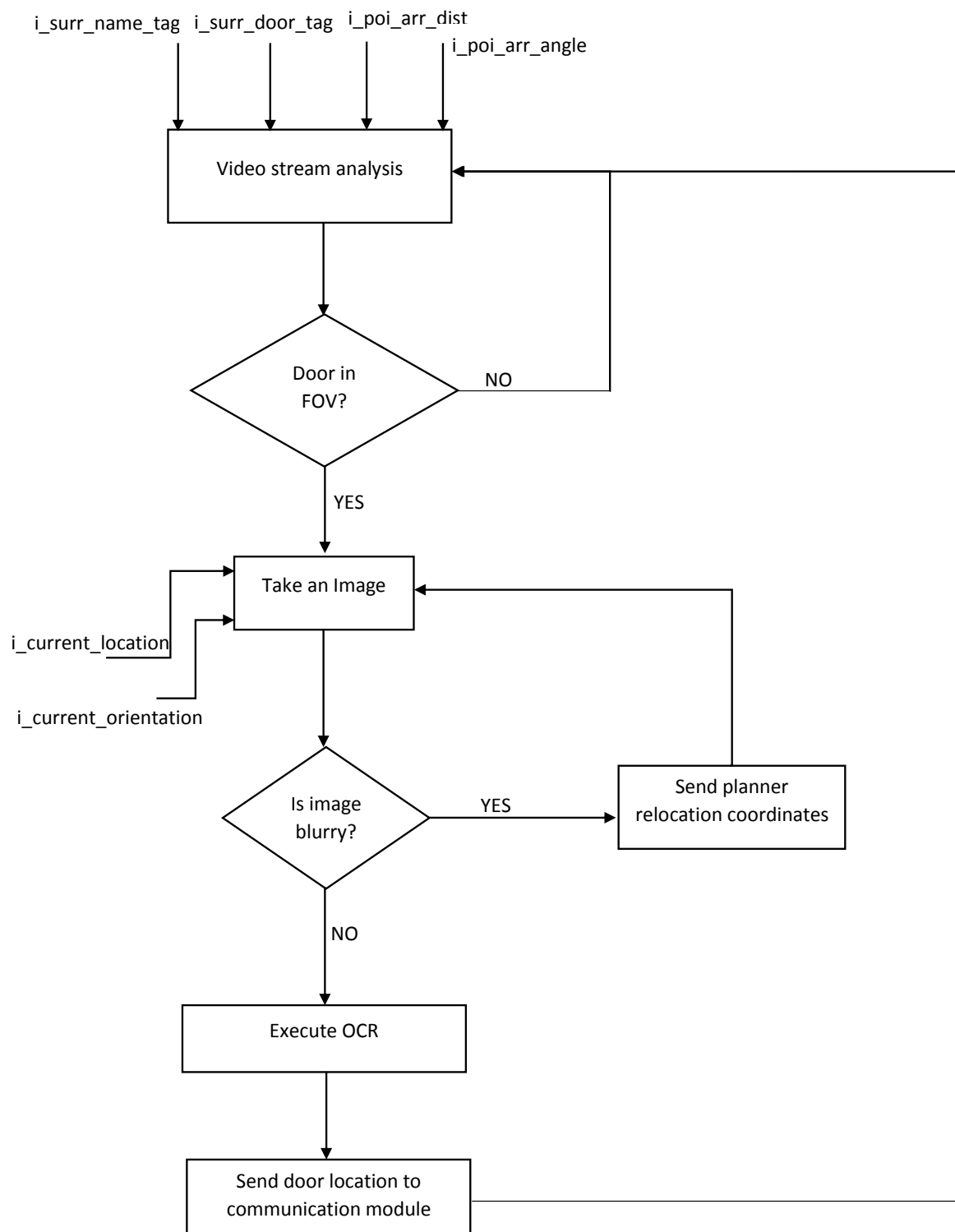
Class: OCR			
Function	Parameters	Return Type	Description
Run	Image	String	Runs the OCR on the image and return the extracted text as a string

Diagnostic_Unit

The current state of the POI module will be reported to the Diagnostic Center. The Diagnostic Center will have no control over any actions performed by POI.

POI Module Implementation Flowchart

The following flowchart will explain the high level process of execution of the POI module.



Algorithms

Implementation of Object_Detection

The paragraph will explain the process of analyzing a video stream to determine if an object of interest is in the camera Field of View. The object detection can be done by filtering out every color except the color of the door and the required tag. Since the tag and the door are the same color, the process of filtering is simplified greatly. By converting the video stream from BGR (Blue, Green, Red) color space to HSV (Hue, Saturation, Value) color space, we can filter out every color except the one required. Playing with the minimum and maximum values of HSV will result in the color required turned into a white pixel and every other color into a black pixel. The following diagram will clarify what happens when filtering for a specific color and having the right



The block on the left is in BGR color space. If we filter so out every color except green, we get something like the block on the right, where the white box represents green and every other color is represented by black pixel.

Determining Image quality

Determining the quality of an image or if an image is blurry is a difficult process. One of the way to determine if an image is blurry is to determine the sharpness of an image. Sharpness can be determined by using the Laplace filter. There are other methods to determine the best possible solution, but all these methods need to be tested with the required hardware to determine the best possible solution for this system.

Implementation of OCR

Implementing an OCR is a simple process. All that is required is the use of an open source OCR engine and just running commands to execute the OCR process. This process will read the contents of the image and write to a text file.

Design Alternatives

- The POI shall contain two USB camera units facing outwards 180 degrees apart from one another
 - **Reason for Rejection:** USB cameras introduce a lot more CPU overhead in comparison to ribbon cable camera.
- The POI shall contain two cable camera units facing outwards 180 degrees apart
 - **Reason for Rejection:** Raspberry Pi board only supports one cable camera unit. To use two ribbon cable camera units, two Raspberry Pi boards will be required. Implementing this solution would give higher rate of success at detecting doors and improve the time required to build the map. But due to budget and space constraints, it is not practical to implement this solution.

Communication Module

Module Requirement Specification

CM_MR_01

The communication module shall be able to buffer messages when connection to the Off-Board modules is down until connection is re-established

Rational: Module is using wireless communication which may disconnect intermittently at which instances communication module shall be able to maintain its function by buffering the messages until the communication is re-established

Fulfill Requirement: SR_FR_4-4

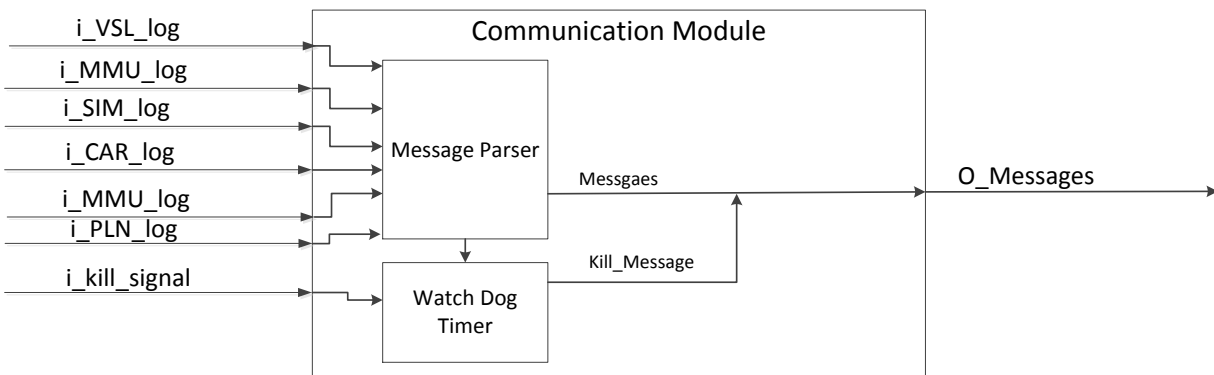
CM_MR_02

The Communication module shall be able to identify when any of the ON-Robot modules are offline or not properly responding and communicating that information to the carrier module

Rational: Carrier has to stop until all modules are on and ready especially if any of the on robot communication hardware is damaged

Fulfill requirement: SR_FR_4-6

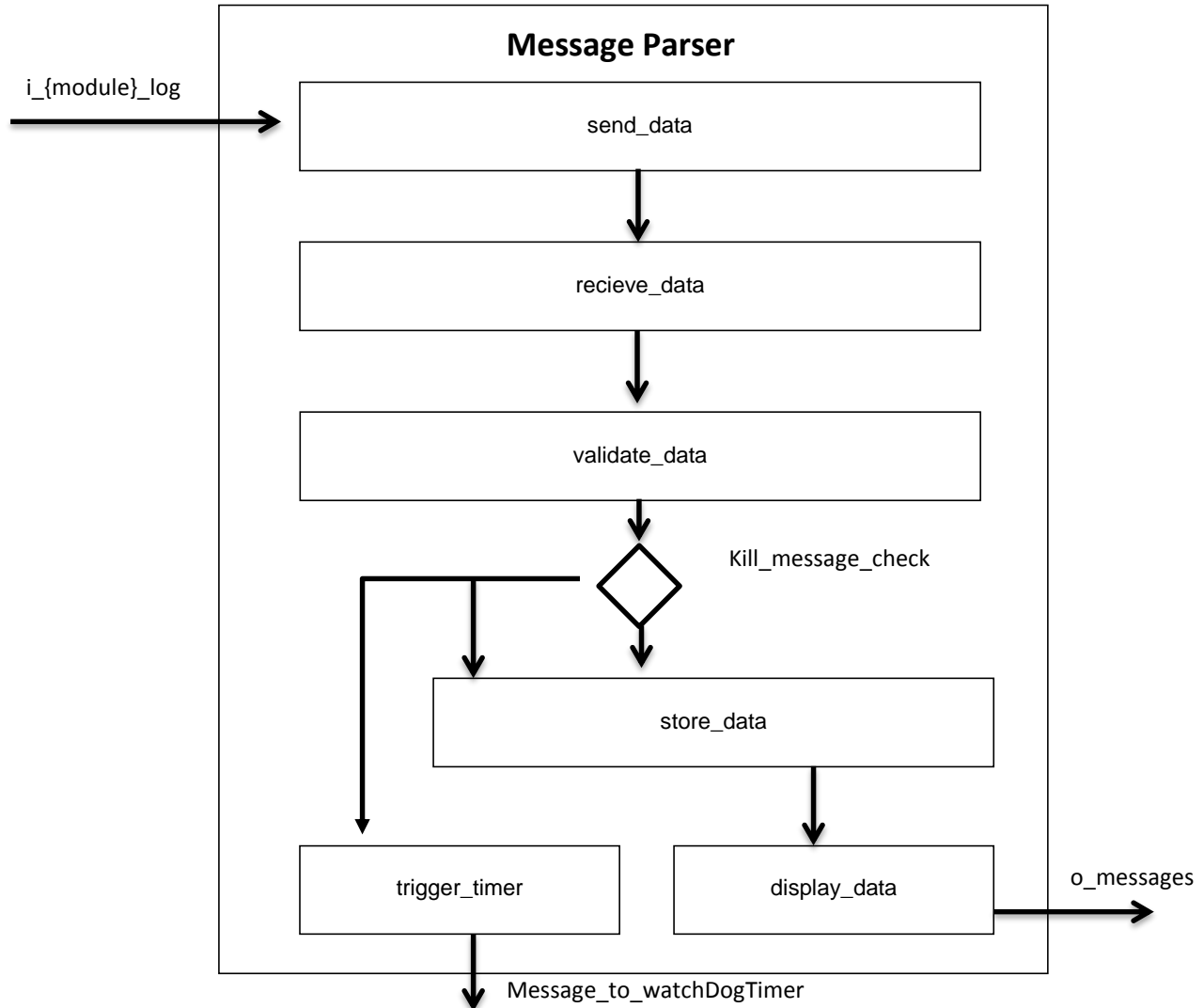
Module Functional Breakdown



Module Design Description (MIS & MID)

Message Parser

This module will be concerned with reading and buffering all the messages from all robot modules that are targeted to the Visualizer or the Command Center.



Function	Parameters	Returns	Description
send_data	string	Int	On vehicle side, responsible for sending all the data from the unit over wifi to the designated receiving point on the off board communication module part.
receive_data	string	int	Off Board side, responsible for receiving the logs and passing them to the validation function.
validate_data	string	int	processes data received the validates it's according to database format, and if a watchdogTrigger is found, it sends an interrupt signal to the timer
store_data	string	int	Sends data to database for storage.
display_data	void	string	Pulls all new data from the database and sends to the command center to be displayed on the appropriate views
trigger_timer	void	int	sends a interrupt signal to the Watch Dog Timer, to trigger the Kill Switch

Watch Dog Timer

This module will be listening to the logs from all the modules to make sure they are responding in time. If a module goes offline or any of the communication streams fail it will try to reach the carrier with an e-Stop (Kill signal) message.

This module is also listening to the kill_signal from the command center if the user decides to shut down the system.

Function	Parameters	Returns	Description
interrupt()	void	Int	When triggered, it'll send the kill switch command.

Communication Module Inputs and Outputs

Input/Output	Variable Names	Type	Description	Range
Input	i_SIM_log	String	Log from the Sensor Interface Module	
Input	i_CAR_log	String	Log from the Carrier Module	
Input	i_MMU_log	String	Log from the Modular Mapping Unit	
Input	i_PLN_log	String	Log from the Planner Module	
Input	i_POI_log	String	Log from the Point of Interest Module	
Input	i_VSL_log	String		
Input	i_CM_log	String		
Input	i_kill_signal			
Output	o_planner_command			
Output	o_POI			
Output	o_log			

Visualizer

Module Requirement Specification

VSL_MR_01

Visualizer shall be wirelessly connected to the vehicle via WiFi

Rationale: Visualizer is off-board and needs a wireless connection to the mapping unit, and WiFi has adequate bandwidth for this project's usage..

Fulfill System Requirements: SR_FR_4-4

VSL_MR_02

Visualizer shall be off-board and run on a computer device (laptop) independent from the indoor mapping vehicle.

Rationale: Users will have the ability to monitor the robot system live remotely

Fulfill System Requirements: SR_FR_4-4

VSL_MR_03

Visualizer shall convert point array sent from MMU to basic (x,y) coordinates to build 2D map and the base point on map shall be the lower left corner.

Rationale: Map data format may change so this part will handle information hiding for the rest of the module

Fulfill System Requirements: SR_FR_3-4

VSL_MR_04

- Visualizer shall construct a visual 2D map using the points sent from the MMU.
 - 2D map shall be (1cm : 1m) scale
 - 2D map shall have indication (astrik) for poi identified.
 - 2D map shall have clickable poi points to lead to further data on poi (name,data,image)

Rationale: Visual map should fit the screen and have functionality for the user

Fulfill System Requirements: SR_FR_4-1

VSL_MR_05

Visualizer shall report its status to the Command Centre module every *k_log_period*.

Rationale: User will be able to monitor the status of the robot

Fulfill System Requirements: SR_FR_4-4

Module Functional Breakdown

The following diagram - Figure 6 - is a high level description of the interactions of the different internal modules of the Visualizer unit. All these components will report to the Diagnostic Center.

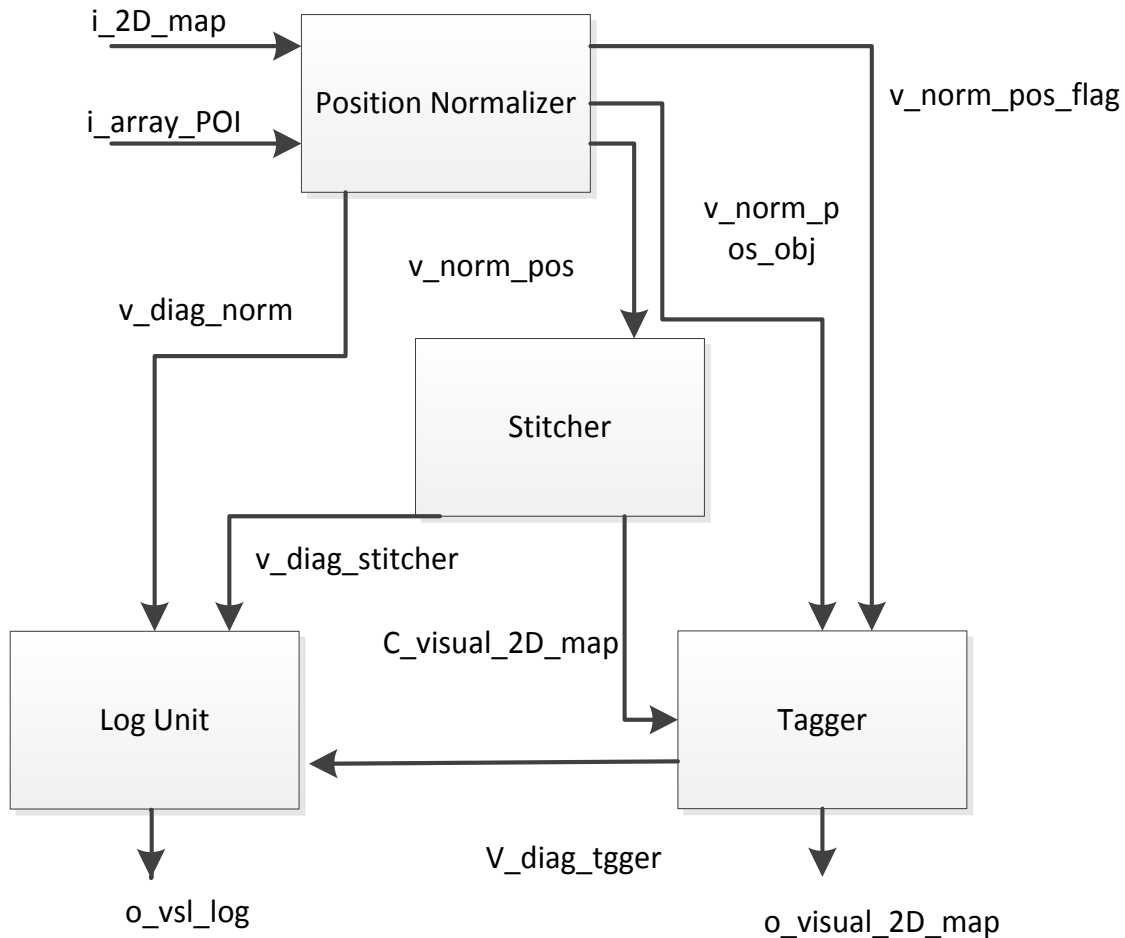


Figure 6 - Visualizer Functional Breakdown

Internal Variables
<i>v_norm_pos</i>
<i>v_norm_pos_flag</i>
<i>v_norm_pos_obj</i>
<i>v_diag_norm</i>
<i>v_diag_stitcher</i>
<i>v_diag_tagger</i>

Visualizer Inputs and Outputs

Input/Output	Variable Name	Description
Input	i_2D_map	Will contain position points from the MMU
Input	i_array_POI	Will contain: position points from POI module matching the MMU points, plus Flag on each point (a 1 or 0) indicating the presence of a POI in the point Object containing image and other captured data regarding poi in point.
Output	o_visual_2D_map	will be visual 2D map of the mapped area.
Output	o_vsl_log	Will contain information about visualizer module for diagnostic purpose

Module Design Description (MIS & MID)

The following is a brief explanation of the main internal components of the Visualizer module:

Position_Normalizer

Takes in the position arrays from the MMU and POI modules and converts them to simple (X,Y) coordinate to be added to the 2D visual map, where the reference point for the map will be the upper left corner

Function	Parameters	Returns	Description
position_normalizer	array of int	Int	Processes inputted data points and corrects any data points to generate an acceptable 2D visual map representation

Stitcher

Generates a visual 2D map by “stitching” together the position points

Function	Parameters	Returns	Description
stitcher	array of int	Int	takes the processed data points from the “position_normazlier” and generates a path connecting all the points resulting in a 2D visual map

Tagger

Uses the generated visual 2D map from Stitcher with the corresponding POI information from Position_Normalizer to add an overlay of POI points and associate each point with its data (name,description..etc)

Function	Parameters	Returns	Description
tagger	array of int	HashMap (key-set values)	generates a key-set value hash map for each poi recognized with its corresponding data point location on the generated path(2D visual map)

Log_Unit

The current state of the various internal module as well as any changes in its behaviour will be reported to the Diagnostic Center. The Diagnostic Center will have no control over the Visualizer.

Function	Parameters	Returns	Description
log_unit	void	string	logs current module states and sends to the communication module (using the “send_data” functions

Command Center Module

Module Requirement Specification

CC_MR_01

Command center will be wirelessly connected to the vehicle via Wifi

Rationale: Since the mapping will be freely moving around and a wireless link is needed to get feedback from the unit

Fulfill System Requirements:SR_FR_4-4

CC_MR_02

Command Center will be off-board, run from a computer device (laptop) independent from the indoor mapping vehicle.

Rationale: Since the mapping will be freely moving around and a wireless link is needed to get feedback from the unit

Fulfill System Requirements:SR_FR_4-4

CC_MR_03

Command center will have to display diagnostic information from the various modules, such as:

- Current module state
- Error Code signals.
- Memory Usage

Rationale: Despite the unit being autonomous, feedback information is needed to monitor system state and perform any necessary debugging steps.

Fulfill System Requirements:SR_NFR_3-2

CC_MR_04

Command Center will be able to send a KILL signal to the vehicle remotely, to stop its modules and movement at any point in time.

Rationale:

Fulfill System Requirements:SR_NFR_3-2

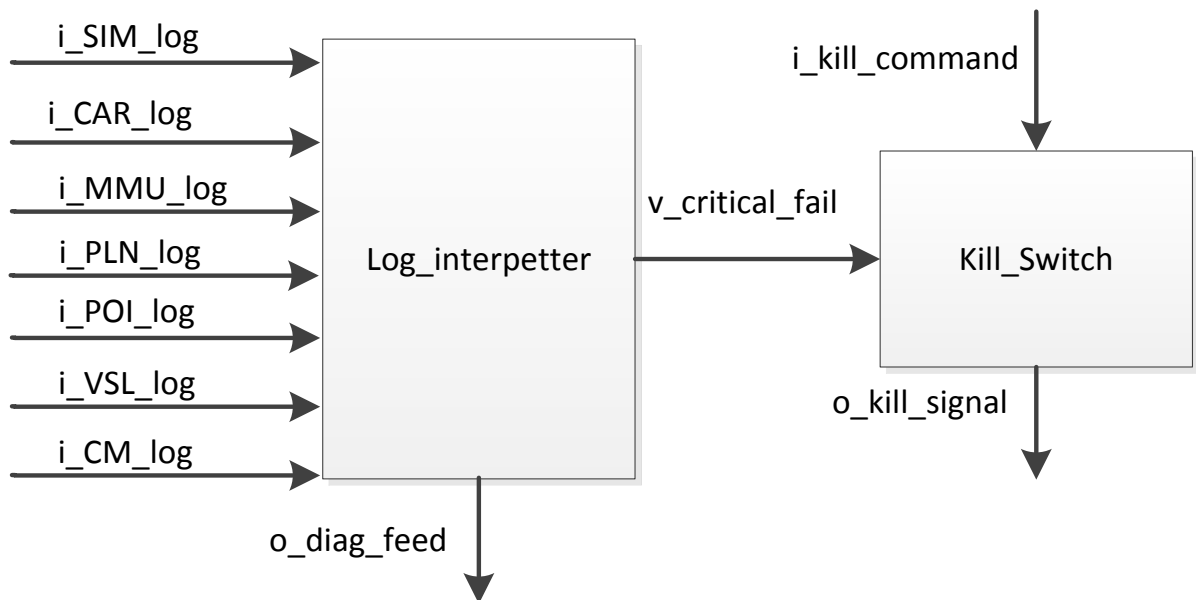
Module Functional Breakdown

Figure 7 - Command Center Module Functional Breakdown

Internal Variables
v_critical_fail

Command Center Inputs and Outputs

Input/Output	Variable Name	Description
Input	i_SIM_log	will contain any diagnostic data from the SIM module
Input	i_CAR_log	will contain any diagnostic data from the Carrier module
Input	i_MMU_log	will contain any diagnostic data from the MMU module
Input	i_PLN_log	will contain any diagnostic data from the Planner module
Input	i_POI_log	will contain any diagnostic data from the POI module
Input	i_VSL_log	will contain any diagnostic data from the Visualizer module
Input	i_CM_log	will contain any diagnostic data from the Communication module
Input	i_kill_command	User Command to kill the system and stop all functionalty
output	o_kill_signal	will send a single command to stop the vehicle and all its functionality
Output	c_diag_feed	Will have diagnostic messages compiled from the different feeds it receives.

Module Design Description (MIS & MID)

Log_interpetter

Takes in the diagnostic feed from all the various modules in the system in the form of logs, and interprets each one and display them to the end user.

The various logs will have the following information in them:

- current module state
- error code
- timestamp

Function	Parameters	Returns	Description
log_interpetter	string	Int	takes in log data from the communication center, processes them and send them to the correct view to be displayed

Kill_Switch

This will be the module, responsible for sending the o_kill_signal command that is responsible for stopping the mapping unit and all its functionality.

Function	Parameters	Returns	Description
kill_switch	void	Int	sends a kill/terminate signal to the vehicle to stop all actions

Normal Operation

Normal Use Case

In a normal use case, the user starts the robot in the hallways of one floor which he wishes to be mapped. The system could be monitored remotely but needs no direct human control. The system autonomously navigates and maps the complete floor, it returns to the initial position not ready and active to be used in navigation assistance.

Please refer to system Requirements specifications for complete list of desired behavior.

Undesired Behavior

Please refer to system design for undesired behavior.

Risk mitigation measures

Risk	Mitigation Plan
Crash	<ul style="list-style-type: none"> E-Stop in planner Emergency Routine continuously monitoring sensors with direct access to low level driver with ability to turn off power to the motors
Loss of communication between modules	<ul style="list-style-type: none"> Watchdog timer with direct access to low level driver with ability to turn off power to the motors
Unforeseen Emergencies	<ul style="list-style-type: none"> Emergency shutdown button on the robot connected to main power source Remote E-Stop switch in Command Center

Division of Labor

Module	Lead Engineer
Visualizer	Abdel-Latif, Sari
Command Center	Abdel-Latif, Sari
Communication Module	Abdel-Latif, Sari
Point Of Interest Module	Batth, Chanderdeep
Planner	ElSaftawy, Mahmoud
Modular Mapping Unit	Mansour, Ahmed
Carrier	Wahid, Fahim
Sensor Interface Module	Wahid, Fahim
Overall System Design	Bishara, Marc