QUICK RESCUE ROBOT (QRBot)

Adwait Bajpai[1], Anand Prakash[2], Ananya Reginald Frederick[3], Jitikantha Sarangi[4]

Abstract: Quick Rescue Bot (QRBot) system involves the surveillance of an area affected by catastrophic disasters e.g. fire, flood, earthquake etc. by using swarm robotics technology. This paper explains the design and working principle of a complex multirobot system by illustrating the abilities of the robots to predict the situation and carry out different roles among themselves according to the hazardous condition they are deployed in. The multirobot system can be described by three application scenarios- Navigation Scenario, Surveillance Scenario and Communication Scenario. The different roles and activities of the robots are specified based on these application scenarios. The Navigation Scenario primarily deals with the movement and stability of the robots, speed and direction adjustment, orientation and alignment among themselves when working in community. Apart from these tasks, the Navigation Scenario also helps the robots detect obstacles and avoid them based on their own intelligence and finally coordinate with each other for a successful navigation. The Surveillance Scenario deals with the detection of living objects by thermal imaging, motion sensors and sound recognition capabilities. Also, it helps in detecting low visibility and taking appropriate measures for the successful completion of survey role. It also maps the underwater ground level to provide a suitable path for the rescue team in case of flood. The Communication Scenario takes care of the communication between the robots starting from the selection of communication interfaces to relaying information among them. This paper provides an insight into the structure, function, task, role, activity and responsibility of each robot in the multirobot system by making use of the Mechatronics UML approach. It describes the role-based behavior[1] and the design implementation of the system software according to the different roles specified. It also describes the implementation of the cognitive operator of the system which deals with running the optimization loop to ensure proper communication among the robots by taking different dependable factors into consideration.

Keywords: ***QRBot, Rescue Assist Multirobot System, Rescue/Surveillance Team, Navigation, Surveillance, Communication, Component, Roles, Behaviors, UPPAAL, Real Time State Chart, OCM***

1 Motivation

When disaster strikes, we rely on first responders to rush into the affected area to save lives and property. Nowadays, the environments and the extent of damage in case of natural calamities and catastrophes are so high that direct human intervention is beyond dangerous.

[1] Department of Computer Science, FH Dortmund, Germany, adwait.bajpai001@stud.fh-dortmund.de
[2] Department of Computer Science, FH Dortmund, Germany, anand.prakash002@stud.fh-dortmund.de
[3] Department of Computer Science, FH Dortmund, Germany, ananya.frederick005@stud.fh-dortmund.de
[4] Department of Computer Science, FH Dortmund, Germany, jitikantha.sarangi001@stud.fh-dortmund.de

In this present era of technology, unlike the traditional rescue operation being carried out, robots are joining the battle to help save human lives. These are called **Rescue Robots**. Well, this paper describes the behavior and functionality of a **Rescue Assist Multirobot System** as shown in Figure 1 which would help the **Rescue Team** carry out the rescue operation in a smooth and efficient manner. This Rescue Assist Multirobot System consists of a set of bots (one leader and two follower bots) with added complex capabilities.

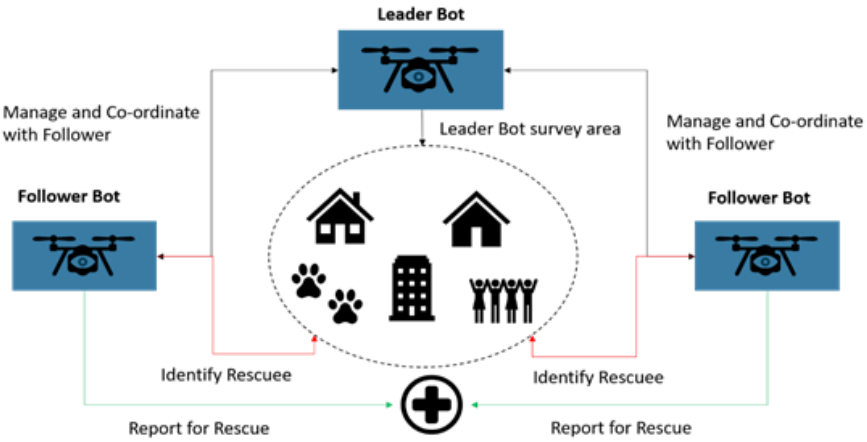

Figure 1: Rescue Assist Multirobot System

2 Goal

In multirobot system, simple robots communicate and coordinate with each other to achieve some well-defined tasks which would otherwise be impossible for an individual robot to achieve. The real power comes when many robots work in a group according to their defined roles and behaviors, so as the complexity. They interact, communicate and coordinate with each other to achieve intricate tasks. It is also a fact that building a multi robot system is cost effective than building a single robot with all the complex capabilities incorporated in it.

The leader robot is responsible for surveying an area and generating 3D map of the rescue area. It coordinates with the follower robots by maintaining distance between each of them and delegating specific rescue assistance tasks. The leader robot is also responsible for sending the data over cloud which can be used for further analysis and cognitive learning of the robot. It also helps the rescue team by identifying the safest, fastest and reliable path to rescue the victim. The role of follower robot is to coordinate with the leader robot and

perform the tasks delegated by it. The detection of victims is performed by the follower robot and the data is relayed to the leader bot.

The bots are considered to perform in cases of fire hazards, floods and earthquakes. In case of fire hazards, the bots could go inside the building on fire and could locate or identify if any human being or animal is stuck or needs help. They would have capabilities to detect living beings through dense smoke, tolerate high temperature because of their rigid and fireproof materials they'd be made of. In case of floods and earthquakes, they could work in communities to survey the affected areas to find out whether any living being is stuck somewhere or not. Particularly, in case of floods, they could do underwater floor mapping to help the rescue team select suitable paths to rescue the living beings as soon as possible. In case of earthquakes, **3D map generation** of the affected building or area is one more important help that they could help the rescue team with.

3 Environment Model

The system comprised of one master or leader bot and two slave or follower bots. Any bot can act as a master or slave at a given point of time which will be explained in detail later in the role section. In the **Block Definition Diagram** (BDD) of the Environment Model, the bots come in direct contact of the environment and the information flow is shown in Figure 2. They would have the capabilities to detect the weather conditions according to the information flow and act in the most suitable way possible. They are also connected to the channel which could be either water or fire or air depending on the area where the rescue operation would be carried out. They could be able to detect the temperature of the channel, the speed and direction of air and water flow and act accordingly. They could also be able to detect living beings and obstacles and avoid obstacles by making use of the smart sensors. The leader bot would be connected to the cloud to successfully transmit and receive real time data. Also, it would be connected to the Rescue/Surveillance Team to execute the deployed commands and to transmit on field real time data.

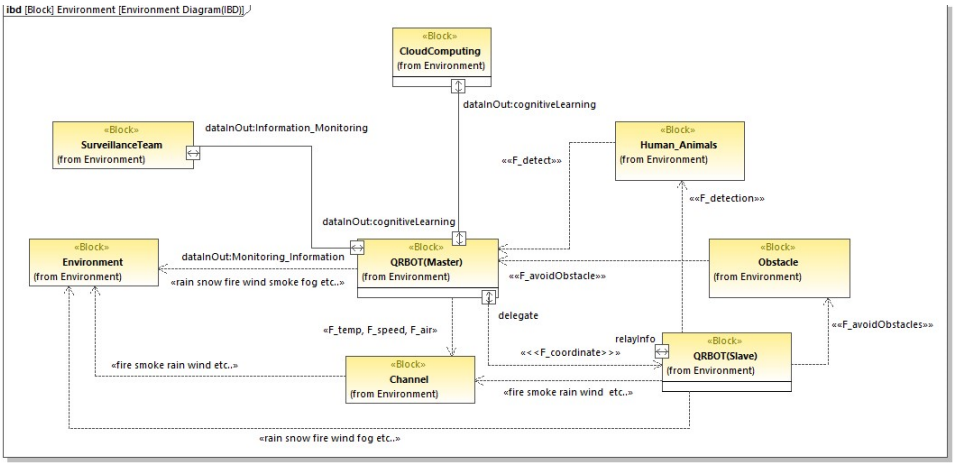

Figure 2: Environment Model

4 Application Scenarios and Principle Solutions

The fundamental working principle of the rescue assist multirobot system has been described by the following application scenarios and for each application scenarios, a set of principle solutions have been defined.

4.1 Navigation

- **Scenario**: The bots must be able to coordinate with each other in order to provide better rescue assistance. The primary task of this scenario is the management of their movement and stability, speed and direction adjustment, proper orientation and alignment among themselves when working in community. They should be stable when in motion. They should be able to adjust their speed and direction based on the channel speed and direction.
- **Solution**: At first, the leader bot does a quick survey of the affected area. After that, it assigns the follower bots with respective locations to carry out the detection task. If, by any chance, one follower bot comes under the zone of another, the leader bot should ensure to arrange them accordingly so that their line of action and angle of view do overlap. All the bots should have the capability to detect obstacles by means of LIDAR and IR sensors and could avoid those obstacles for smooth movement.

4.2 Surveillance

- **Scenario**: The rescue bots must act in a community and each bot must be assigned specific tasks. They must be able to detect living beings to be rescued and do the survey of the area under rescue.
- **Solution**: After the leader bots assigns the follower bots the respective areas for the survey and detect operation, the follower bots start their work. They could have the capability to detect objects via thermal imaging, sound sensors and motion sensors. They should also have the capabilities to detect objects under **low visibility** condition. They should have the capability to do **underwater floor mapping** via ultrasonic sensors so that they could help the rescue team with selecting a suitable path to reach the rescue destination easily and in less time.

4.3 Communication

- **Scenario**: The bots must have the capabilities to identify the best communication interface in order to communicate among themselves as well as the rescue team.
- **Solution**: Initially, a leader bot is selected. All the follower bots must be registered with the leader bot at first. In every bot, there should be a table containing the IDs of all the bots. If the leader bot fails due to some reason, a new leader bot would be selected based on these IDs. The signal strengths of the communicating devices of the bots are analyzed at real time. If the signal strength is week, an optimization loop should be run to enhance the signal strength or hop to some other communication interface as suited.

5 Requirements

A set of requirements has been defined for the design and implementation of the system. These requirements are divided into three classes i.e **structural requirements**, **non-functional requirements** and **functional requirements**.

5.1 Structural Requirements

The requirements which define the overall structure, geometry and appearance of the bots are listed in Table 1.

Table 1: Structural Requirements

Requirement ID	Requirement Specification
1.1	Dimension of each bot must be within 60cm*50cm*25cm.
1.2.1	Each bot should consist of 2 wings i.e. left and right.
1.2.4	Elliptical from top and floating architecture at bottom.
1.3	Weight of each bot must not exceed 2 Kilogram.
2.1	The design of each bot must ensure stability in air.
2.2	The design of each bot must ensure buoyancy in water.
3	The material used to make the bots must be able to handle harsh weather conditions and withstand wide range of temperature.

5.2 Non-Functional Requirements

The requirements which define the registry mechanism, leader and follower bot selection and power management of the bots are listed in Table 2.

Table 2: Non-Functional Requirements

Requirement ID	Requirement Specification
7	Uninterrupted power supply must be provided to each subsystem of each bot.
7.2	Each bot must have the capability to detect low power if the battery capacity falls below 15%.
7.3	The primary power supply of each bot should be provided by a lithium ion battery.
7.3.1	Each bot must have full charging capability within an hour from zero charge.
7.3.2	The mass of battery must not exceed 700 gm.
7.3.3	The temperature of the battery must not exceed 70 degrees.
7.4	The auxiliary power supply is provided by solar panel.
7.5	Once fully charged, each bot should be able to operate for a minimum of 3 hours.
16	Each bot must have a unique bot ID for identification.
16.1	At the time of startup, bots should look for a master bot. If there is no master bot available, a master is chosen based on the IDs of the bots.
16.2	Each bot maintains the bot ID table provided by the master bot.
17	Master bot would have an identification number of 255.
18	In case the master bot fails or runs out of power, a new master should be assigned based on the bot ID table.
18.1	After a new master has been selected, its ID changes to 255, bot ID table is updated and broadcasted to all the follower bots. In every follower bot, the bot ID table should be updated.

5.3 Functional Requirements

The requirements which define the fundamental working principle of the system are listed under Functional Requirements. These requirements are divided into three categories based on the application scenarios i.e. Navigation Requirements, Surveillance Requirements and Communication Requirements.

- **Navigation Requirements**: The requirements listed in Table 3 are the fundamental aspects dealing with the Navigation scenario as specified in Subsection 4.1.

Table 3: Navigation Requirements

Requirement ID	Requirement Specification
9	Follower bot must be able to do underwater floor mapping.
9.1	Follower bot must have the intelligence to determine the depth of water.
10	Bots must have the capabilities to detect or sense obstacles and act accordingly.
10.1	Bots should have the capabilities to avoid obstacles.
11	Automatic speed and direction adjustment according to the surrounding conditions.
12	Bots should have the capabilities to accelerate and decelerate.
12.1	Bots should move in water and air using thrust and rotor respectively.
12.1.1	The thrust and rotor speed must be adjusted based on the intensity and direction of the flow of water and air.
12.2	Acceleration and deceleration should happen using the rotor speed of the wings.
12.2.1	Bots should be able to hover in air.
13	Bots must be able to align themselves based on spaces and detected obstacles.

- **Surveillance Requirements**: The requirements listed in Table 4 are the fundamental aspects dealing with the Surveillance scenario as specified in Subsection 4.2.

Table 4: Surveillance Requirements

Requirement ID	Requirement Specification
8	Bots must have the capabilities of surveillance in multiple operating conditions and hazards.
8.1	Bots must have the intelligence to detect living things.
8.1.1	Bots must be able to identify humans.
8.1.2	Bots must be able to identify animals.
8.2	Bots must be able to identify open spaces to assist the rescue team.
8.2.1	Bots should be able to generate 3D map of the area.
8.3	Bots should be able to identify the voice/sound of the living beings to be rescued.
8.4	Bots should have the intelligence to detect entry and exit points.
8.4.1	Bots should have the intelligence to understand sign boards.

- **Communication Requirements**: The requirements listed in Table 5 are the fundamental aspects dealing with the Communication scenario as specified in Subsection 4.3.

Table 5: Communication Requirements

Requirement ID	Requirement Specification
4	Bots must have the capabilities to provide their locations.
4.1	Bots must be equipped with GPS devices.
4.1.1	Bots must be able to send GPS signals and coordinates.
4.1.2	Bots must be able to receive GPS signals and coordinates.
5	Bots should support multiple channel interfaces.
5.1	Tx/Rx antennas must tune to agreed frequencies for data transfer between devices.
5.2	IoT nodes are to be used for high speed data transfer where network connection is available.
5.3	Bots must be able to communicate the long distance receiver e.g. deep sea exploration.
5.4	The leader bot must communicate to cloud to store data for future use and investigation.
6	The system should support high quality data transfer.
6.1	The communication channel must have high bandwidth to support fast data transfer.
6.2	Bots must be able to store data in case no communication is established.
15	Bots must be able to communicate with each other.
15.1	The leader bot must be able to communicate with slave bots and the rescue team.
15.1.1	The leader bot must be capable of establishing control and coordination among the follower bots.
15.1.2	The leader bot must delegate tasks to the follower bots.
15.2	The follower bots must have the capabilities to relay information only to the master bot.
15.2.1	The follower bots must register to the leader bot during start up.
15.2.2	The follower bots must send data and information to the leader bot.

6 Macro Architecture

Figure 3 shows the Macro Architecture of the QRBot. The composition is represented in form of Leader and Follower Bots. Structure and geometry of the leader and the follower Bots are same, functionally they are different. The reliability of the rescue operation is dependent on the reliability of the following:

- **Navigation**: The main objective of the navigation scenario is to help the bots navigate in the environment they are deployed in. This includes collecting surveillance data from surveillance subsystem and processing them accordingly. Data received from Lidar, Motion Sensor and IR Sensor are processed and computations are performed for

Thrust and Rotor Controller. These thrust and rotor controllers actuate the mechanical components (e.g. rotor), which in turn change the trajectory of the Bots. This also involves the process of obstacle detection and avoidance. In this way, the Navigation scenario helps the bot successfully navigate through its environment.

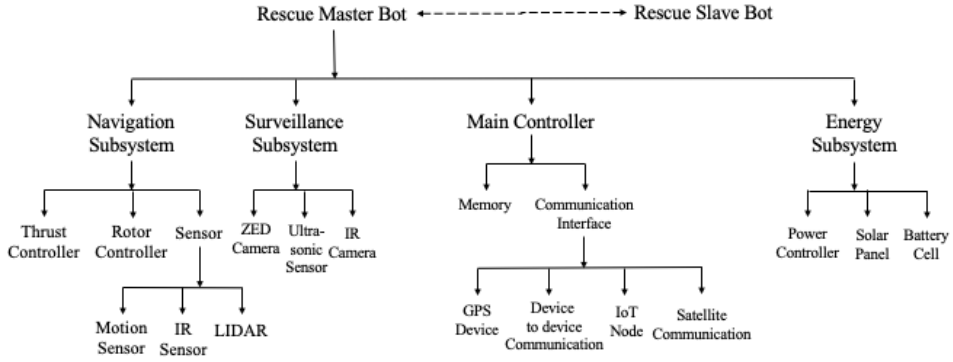

Figure 3: Macro Architecture

- **Surveillance**: The main objective of surveillance subsystem is to survey the environment and collect as much data as possible. This shall enhance the quality of the decisions made in rescue operations. The data collected is operated on by the main controller. The data collected is used to classify objects, living and non-living things, and to represent a clear picture of the affected area to the rescue team. This shall keep the rescue team well informed about the environment and improve the timeliness and quality of response. The surveillance subsystem is composed of the following components: ZED Camera, Ultrasonic Sensor and IR camera.

  ZED camera is used for real time 3D map generation. Ultrasonic sensor is used for measuring distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. IR camera detects and measures the infra-red energy of objects. The camera converts that infra-red data into an electronic image that shows the apparent surface temperature of the object being measured. It can work even complete darkness giving the bot another important functionality- to be able to work in low visibility conditions. These components play a crucial role in rescue operations. Satellite Communication is to be used in deep sea applications.

- **Main Controller**: The main controller is tasked with coordinating activities among different subsystems of the bot. It also has memory to store the data in case of communication failure. It is also comprised of the Communication module. The communication module is an interface to the following communication and positioning technologies:
  GPS, **D2D Communication** (Device to Device Communication), IoT and Satellite communication.

The GPS Device keeps the location updated for each of the bots. This helps in distance coordination among the bots when operated in leader-follower configuration. D2D communication is used in case of floods where telecommunication networks are disrupted. This shall help the transmitters and receivers synchronize over a chosen frequency band. In case of Fire, telecommunication networks are operational, and the leader-follower coordination takes place over allotted telecommunication frequencies.

- **Energy Subsystem**: This subsystem supplies the serried amount of power to the subsystems. It is comprised of a power controller, that regulates power as per requirement. It is equipped with safety mechanisms to prevent over current supplies, and damage to other subsystems. Under normal scenarios, power is supplied by battery cells. For deep sea applications, there is a provision for solar cell extension. The solar panel is also capable of supplying auxiliary power to the bot whenever necessary.

7 Functional Model

Figure 4 is a Functional Model of Inter-bot communication. It depicts the functions performed by the main controller and what the main controller is composed of. Communication takes place between bots for the leader to Delegate tasks to the followers, coordinate tasks and relay information to the followers. As the master indirectly controls communication, it has functions viz. sendData(), rcvData() and storeData().

Figure 5 is a Functional Model perspective of the Communication physical layer interaction. The functions performed by the leader bot include setting communication parameters of the followers viz. Bit Rate, Baud Rate, Data/Channel Encoding and decoding, Error Detection. All the functions are performed by the main controller, which has a communication interface.

Figure 6 is a depiction of Surveillance functional model. The surveillance subsystem is composed of ZED camera, IR Camera, Ultrasonic Sensor and audio sensors. These sensors detect humans, animals and sound. The sensors along with data processing procedures create an exact replica of the environment in which the bot is deployed. Surveillance subsystem also creates the 3D map which is used by the rescue team in rescue operations.

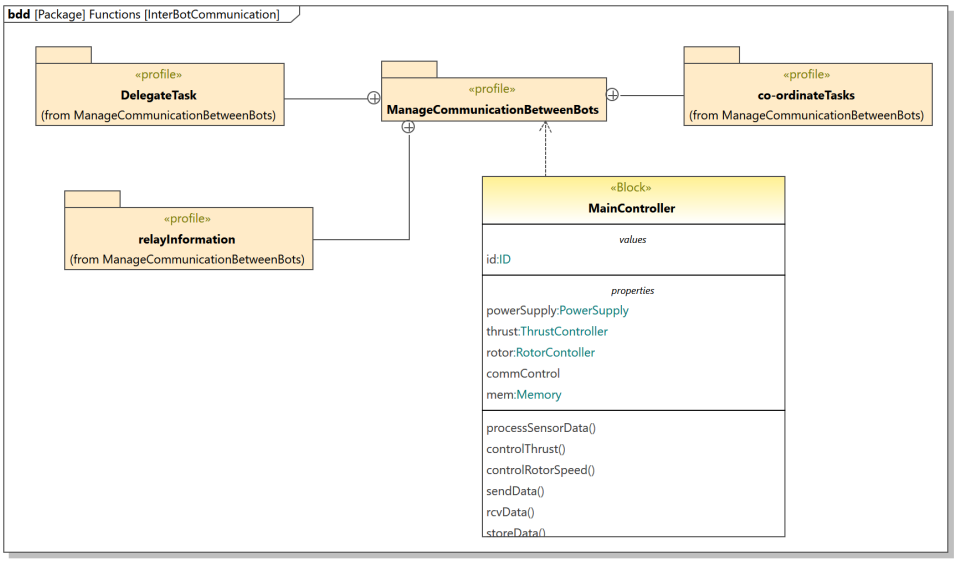

Figure 4: Inter-bot Comunication Function

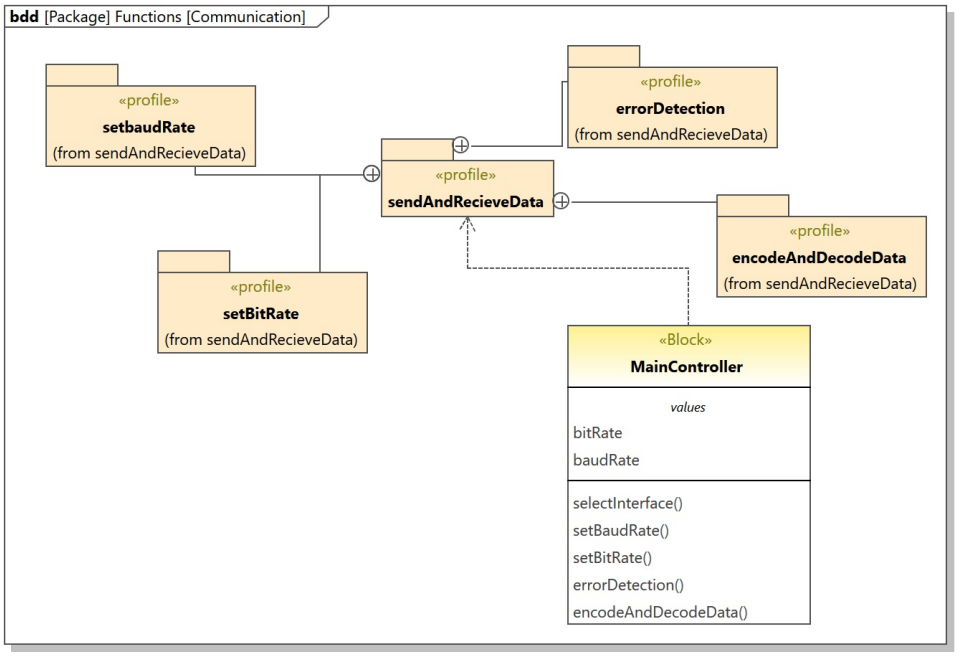

Figure 5: Comunication Mechanism Function

Figure 7 is a functional model of obstacle detection. Obstacle detection is the process of using sensors, data structures, and algorithms to detect objects or terrain types that impede motion. This includes sensor data collection and filtering, identification and classification of the 3D points, cluster those points accordingly to know whether these points all-together correspond to an obstacle or not and finally measuring the distance of the bot from the obstacle.

Figure 8 is a functional model of the Energy Subsystem as described in Sect. 6. Its main function is to provide power supply to the bot which includes energy storing, maintenance of uninterrupted power and temperature monitoring etc.

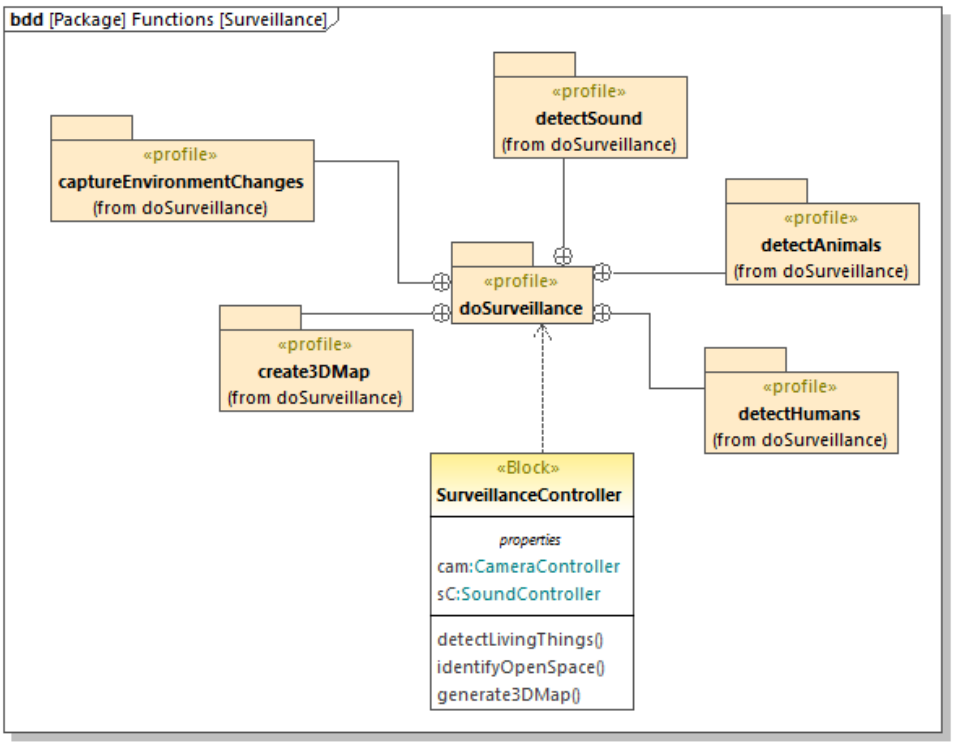

Figure 6: Surveillance Function

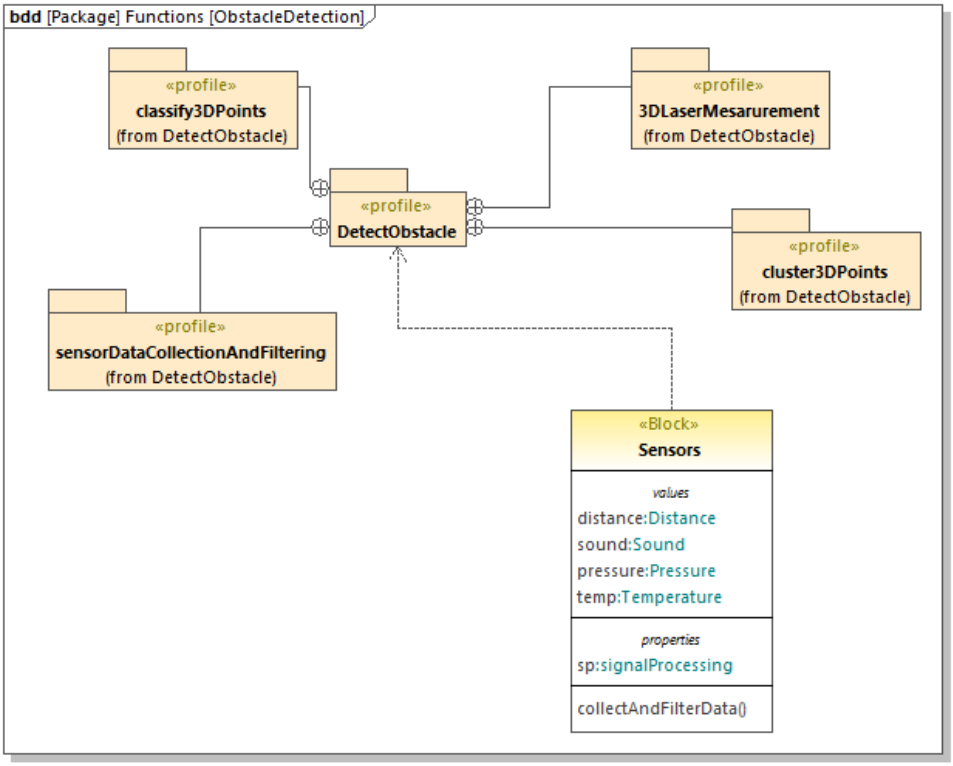

Figure 7: Obstacle Detection Function

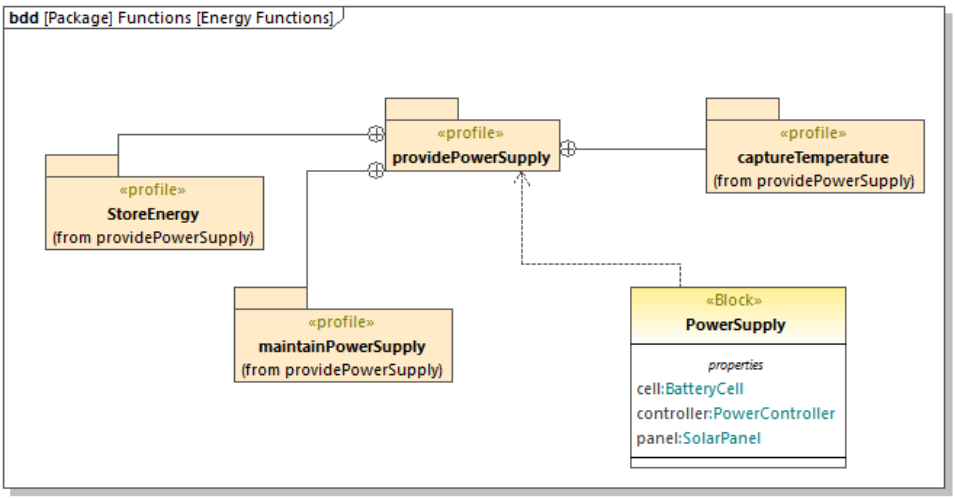

Figure 8: Energy Function

8 Active Structure

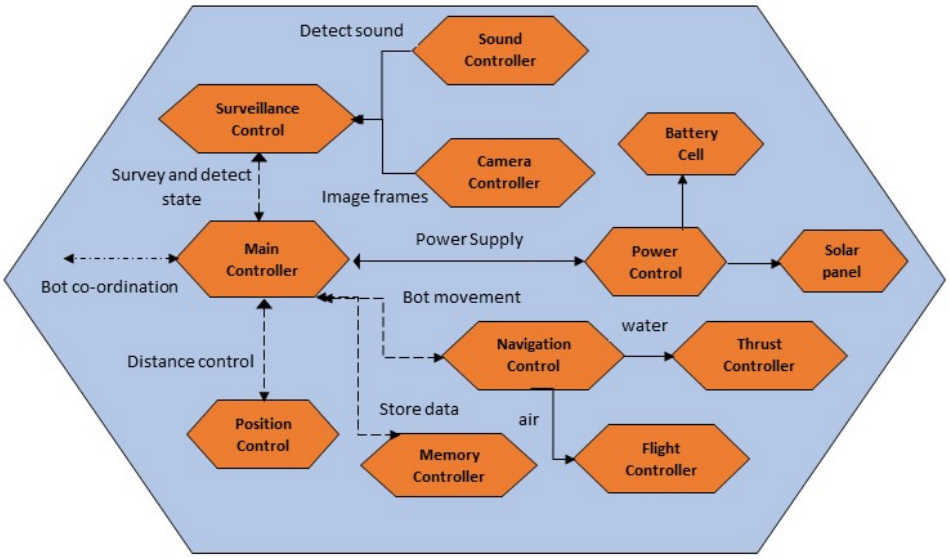

Figure 9: Active Structure

Figure 9 shows the Active structure which depicts the composition of the bots with regards to sensors deployed, flows and allied parameters. Every instance of the Bot is equipped with power supply, surveillance sensors, navigation sensors, thrust and rotor controllers, memory for data storage and algorithms for sensor signal processing. Parameters associated with rotor controller are rotor speed, supply power and wind pressure. Surveillance controller is composed of sound and camera controller. Camera calibration is important in perception and so are the parameters Camera Resolution, Shutter Speed and camera frame rate. Power Controller supplies current and voltage at desired limits. Depending on the operation being used, it selects the power source between batteries and solar panel. Memory used for data storage has parameters like memory size and access speed. All these sensors, along with the parameters form the active structure of the QRBot.

9 System of Objectives

9.1 External System of Objectives

- **Environmental Conditions**: Since the bots would be used in harsh areas and area struck by natural calamities, they would be constantly exposed to the forces of nature which would lead to abrasion in them. Therefore, this needed to be taken into account so that there is minimum abrasion.
- **Cost**: The cost per unit of the bot and maintenance of the bot is another important aspect and the objective is to minimize per unit cost and maintenance cost.

9.2 Inherent System of Objectives

- **Surveillance**: Surveillance is the major purpose of the bots and therefore, important objectives were to maximize the monitoring of the environment and object detection while considering the speed and stability of the bots.
- **Data Transfer**: The surveillance data collected by the bots should be sent back to the rescue team securely while maintaining the quality of the data. Hence, the Data Security was maximized and Interference and Error Rates were minimized. The cognitive Layer of the bot deals with optimizing the transfer of data so that the process is smooth. Implementation of the cognitive layer has been shown in Sect. 12.
- **Energy Efficiency**: Energy efficiency is a critical requirement for the bots and it is an important objective to minimize energy consumption as the bot will need to cover a large area to collect surveillance data and also inform the rescue team of open areas where they can set up intermediate rescue facilities.

9.3 Internal System of Objectives

Abrasion(min), cost(min), speed(min), energy consumption(min), interference(min), maintenance(min), error rates(min) and monitoring environment(max) are selected as the Internal System of Objectives. Using these objectives, the correlation matrix has been plotted.

9.4 Correlation Matrix

The correlation matrix shows the relationship between the objectives. As shown in Figure 10, the relationship between speed of bot and the cost is in conflict with each other which means they influence each other in a negative way. Higher the speed more will be the cost of energy used. Conversely, the energy consumption and cost influence each other directly, that is, if energy consumption is minimum cost will be minimum too. Finally, objectives interference and cost have no influence on each other and therefore it has been given a value of 0. The relationships between all the other objectives can be seen in Figure 10.

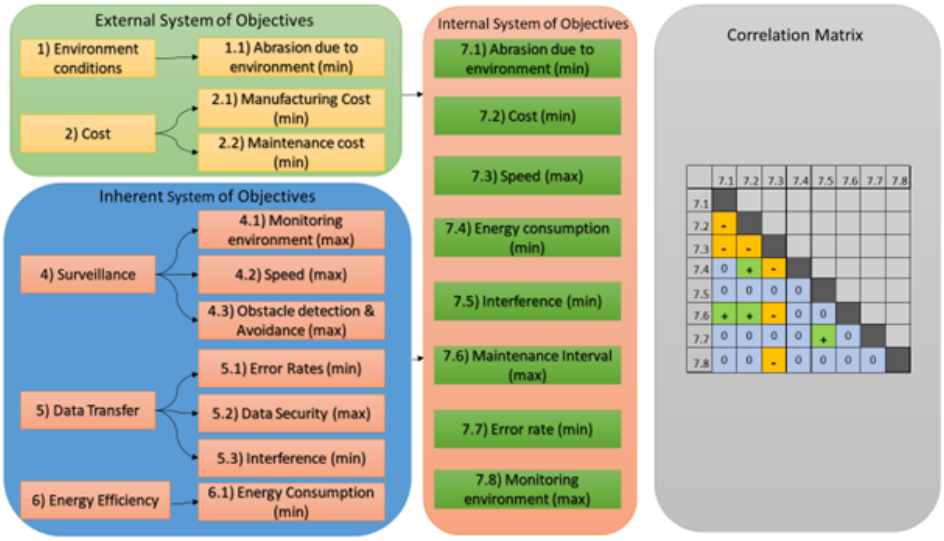

Figure 10: System of Objectives

10 Operator Controller Module

Operator Controller Module is composed of a Cognitive operator, Reflective operator and Controller. Figure 11 shows a clear distinction between the functions performed by each layer. The controller layer is composed of position controller, camera controller, sound

controller and sensor controller. The main operation of this layer is to collect data and send it to the reflective operator. The above-mentioned controllers collect position, image/video, sound and other data useful for the navigation and surveillance operation and send it across.

Information processing happens on the Reflective operator layer. Navigation controller takes input from the position controller and process it accordingly to define the position and movement of the bots. Surveillance controller takes input from the sensor, sound and camera controllers and process all the data to carry out a successful survey of the environment the bots are deployed in. Main controller is the backbone of the complete system. It controls and manages all the controllers and carry out the data flow, calculation and coordination among each component.

Cognition performed to improve system performance is a part of cognitive layer. Behaviour based and Model based self-optimization take place in cognitive part. These optimizations change system parameters at run time to keep up with data quality requirements. Here, the optimization module is the main component which interacts with the main controller present in the Reflective operator layer and the cloud to deal with the data on a real time basis.

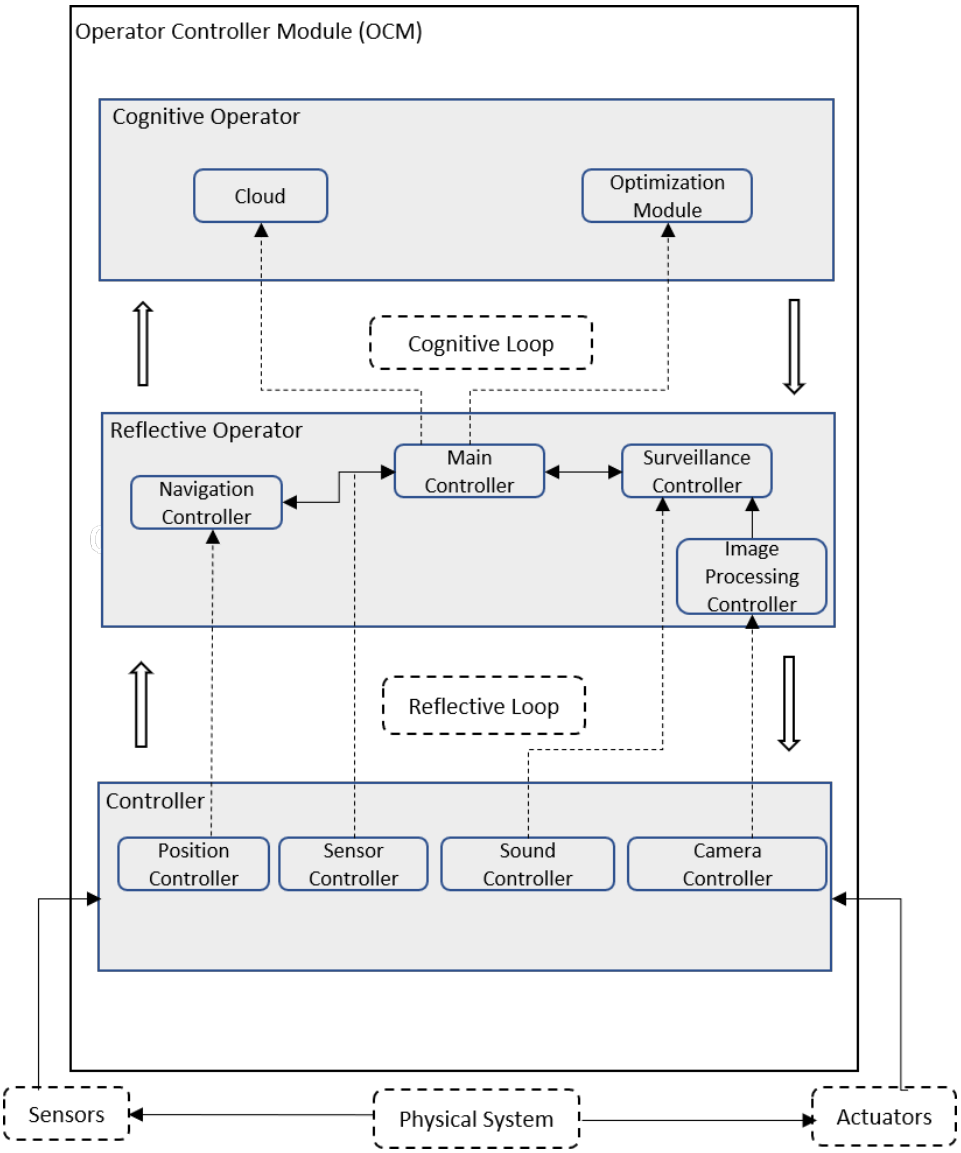

Figure 11: Operator Controller Module

11 Software Design

Mechatronics UML methods are used to design the software for the proposed rescue assistance robot system. In this method, component models are derived from the Active Structure as mentioned in Sect. 8. Each component model is refined into several subcomponents based on the behavior. This reduces the complexity of a single component which, in turn, enables the reuse of existing components and make their verification more efficient[1].

Each component defines a system functionality and they interact only via well defined interfaces called ports[1]. The interaction between two component models are done via ports which is specified by **Modal Sequence Diagram** (MSD). The communication behavior is specified by using **Real Time Coordination Pattern** (RTCP). RTCP defines the required communication between two partners which are called roles. Roles are associated with the well-defined interfaces called ports in the component model. If a role instance of one component model interacts with multiple roles of another instance, we call it a *multirole*.

Now that we have established the component model, Modal Sequence Diagram and Real-Time Coordination Pattern, the next step is to define the real time behavior of each role. This is done using **Real Time State Chart** (RTSC). RTSC is a combination of UML state machines and timed automata. The real time behavior of the overall system is then verified on tools such as **UPPAAL** to ensure that the system is deadlock free and fulfills the requirements[2][3].

11.1 Component Model

There are five components identified in the proposed robot rescue system based on the Application scenarios Navigation and Surveillance defined in Sect. 4. These component models are defined based on the requirements specified in Table 3 and Table 4 and the respective function model.

- **Leader Component Model**

  The Leader component model defines the leader role of the system. This role acts a multi-role as it interacts with two follower roles. This role is activated only in case of the robot which acts as a Leader. Figure 12 depicts the component model of the Leader role and its interaction with multiple follower roles. It is responsible for obtaining the position of the follower robots and getting the health information of the follower robots. It maintains a registry table of the follower robots. The navigation controller

is responsible for the movement of the robot itself. The subcomponents *Coordinate_Distance* and *Analyse_Health* operates in the Reflective layer for maintaining the distance and doing the health analysis of the follower robots.

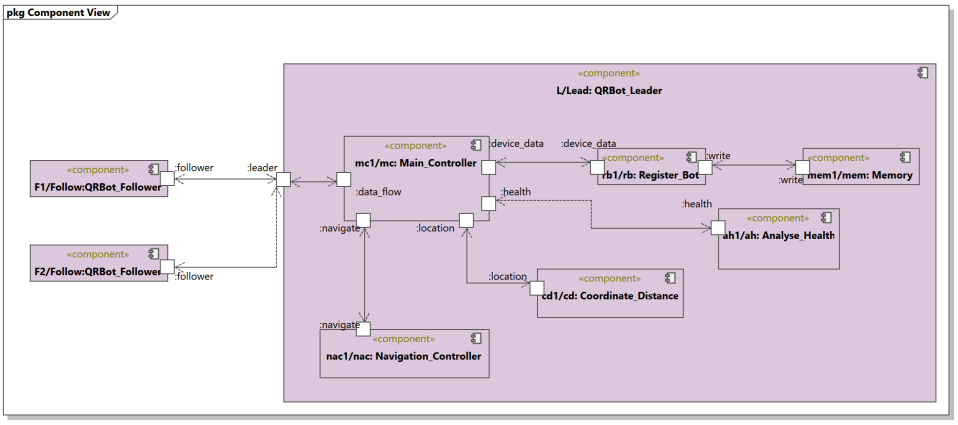

Figure 12: Leader Component Model

- **Follower Component Model**

The Follower component model demonstrates the interaction of follower role with other components. The follower role functions as a single role. Each follower role has a one-to-connection with the Leader role. The objective of follower role is to relay the position information and health information of the devices on the robot to the Leader robot. The subcomponents Position Controller and Device health controller operates in the reflective layer. The information about the position and device health is received from the respective Controller modules. Figure 13 shows the interaction of follower role with the leader role.

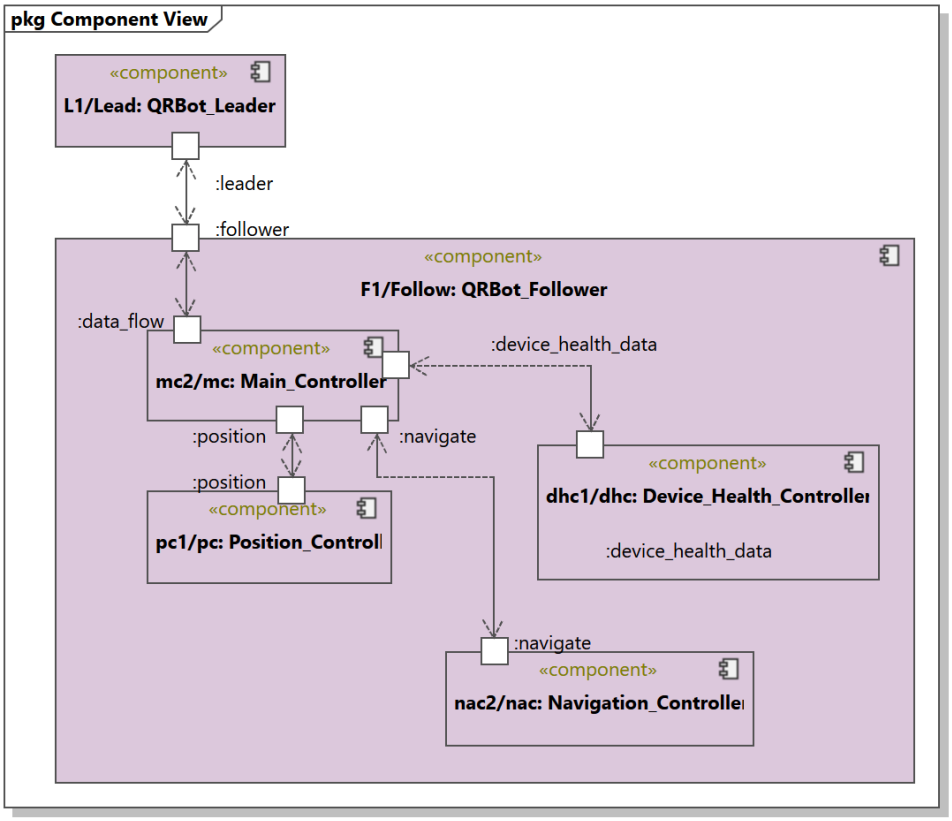

Figure 13: Follower Component Model

- **Survey Component Model**

  The survey component is responsible for doing the survey of the rescue area and conforms to the Application Scenario Subsection 4.2. The survey role interacts with the environment and process the camera images to generate the 3D map of the area. This role is performed by the Leader robot. This helps is assessing the fastest and reliable path to the rescue area. Once the survey of the area is done, it identifies the location where the rescue detection should occur. These locations are shared with the follower robot to detect the victims in those areas. The subcomponents *Image Processing Controller* and *Position Controller* are used for getting the survey information. Refer Figure 14 for the component model view of *Survey Role*.

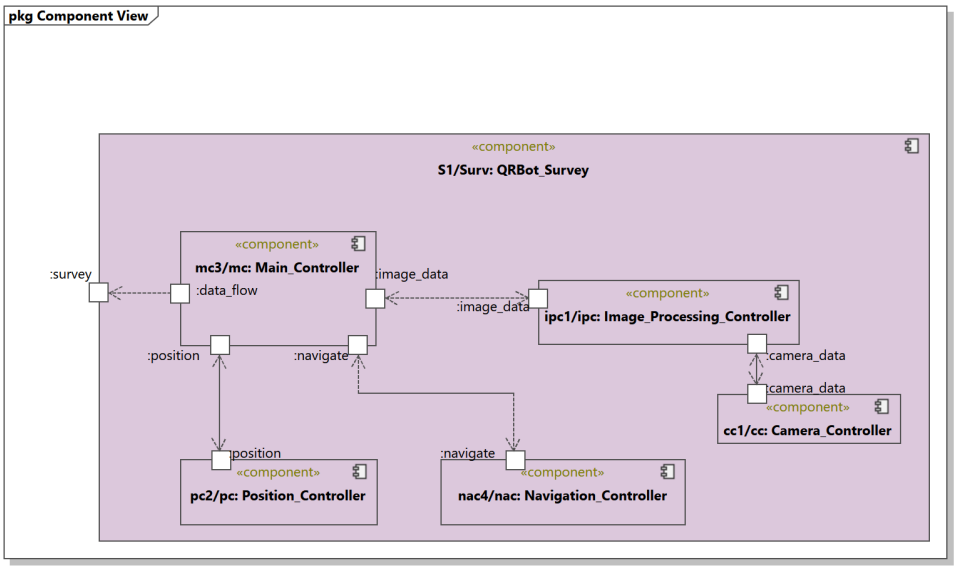

Figure 14: Survey Component Model

- **Detect Component Model**

  The Detect component is responsible for performing the detection operation to identify victims in the rescue area. The detect role uses *Sensor Controller* and *Sound Controller* subcomponents to detect the victims. The number of victims to be rescued is identified based on the thermal imaging data. This information is then relayed directly to the rescue team .This role is performed only by the Follower robot. Refer Figure 15 for the component model view of *Detect Role*.

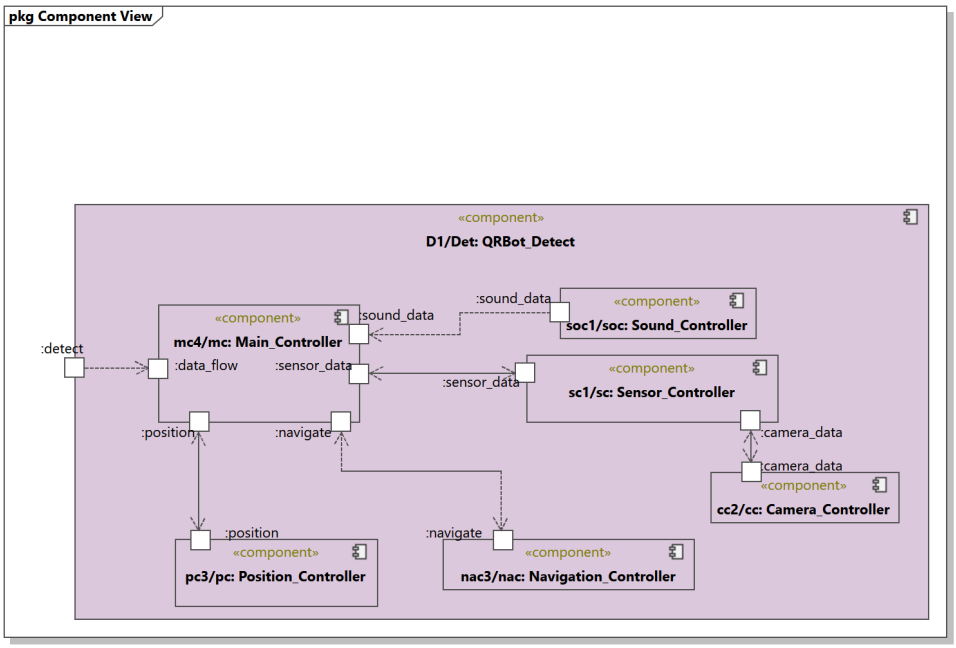

Figure 15: Detect Component Model

- **Coordinator Component Model**

  Figure 16 shows the Coordinator component view of the Rescue Robot. The coordinator role forms a mesh network with the other coordinator roles in the Rescue assistance system, therefore it acts as a *multirole*. This ensures proper synchronization between a leader robot and follower robot. The system is designed such that the role of a robot can be switched from a Leader Robot to a Follower robot. If a switch in the role occurs, all the other robots must be synchronized, and the registry table must be updated accordingly. This functionality is carried out by the coordinator role. There are no specific subcomponents associated with this role. The main purpose of this role is to activate the Leader Role and Survey Role in case the robot functions as Leader Robot or activate the Follower Role and Detect role in case the robot functions as a Follower Robot.

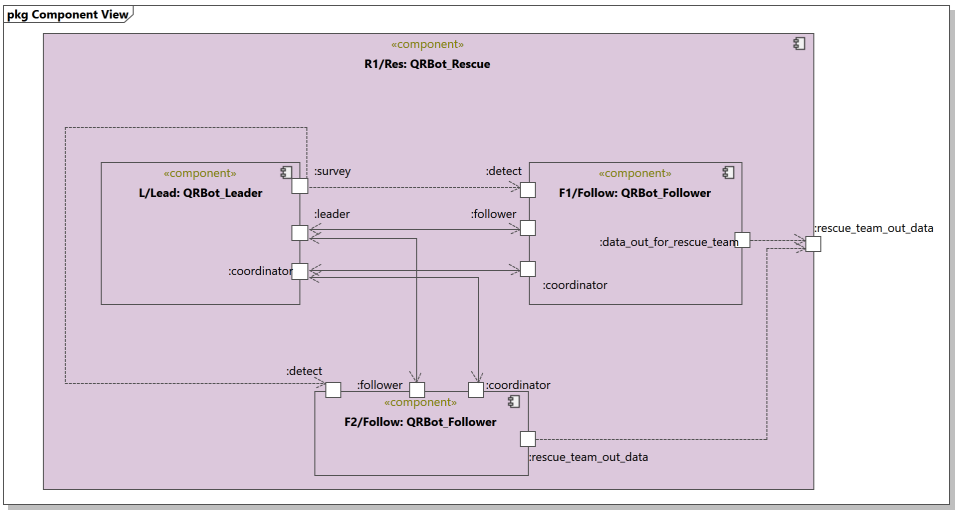

Figure 16: Coordinator Component Model

11.2 Modal Sequence Diagram

The Modal Sequence Diagram demonstrates the sequence of operations that are carried out between the actors to achieve a task. It captures the interaction between objects in context of a collaboration. Figure 17 and Figure 18 shows the sequence diagram demonstrating the interaction between the leader and follower robots in a rescue assistance system.

Initially when the system boots up, each robot establishes a 1:1 connection with other robots. The robots exchange their IDs which is updated in their Registry table. Here the Master Robot is synonymous with the Leader Robot and Slave Robot is synonymous to Follower Robot. The Leader robot is decided among the connected robots followed by updating the registry table entry. The ID of the Leader robot is sent to the rescue team. Figure 17 shows a section of sequence of events that occurs between the actors. The Leader robot sets the GPS position and angle of view on the follower robot in order to perform its activities.

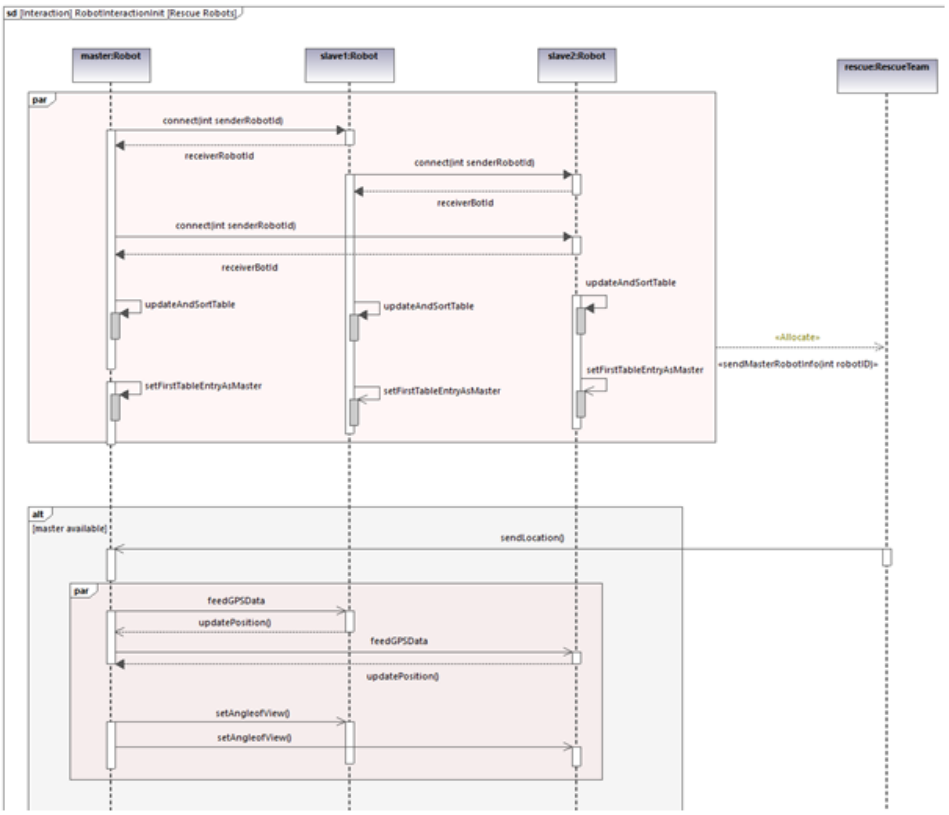

Figure 17: Sequence Diagram showing interaction between the actors when Leader Role and Follower Role is active

The Leader robot analyzes the environment and generates the 3D map of the survey area. Any emergency tasks which is needs to be performed at high priority is sent by the rescue team to the Leader Robot. The Leader Robot can perform the task itself or delegate it to one of the follower robots. If there is any higher priority task requested by the rescue team, the leader robot can delegate the tasks to follower robot as shown in Figure 18. The task for the detection of victims in performed only by the Follower robots. After the detection is complete, the Follower robot sends the data to the rescue team and updates its status to the Leader robot.

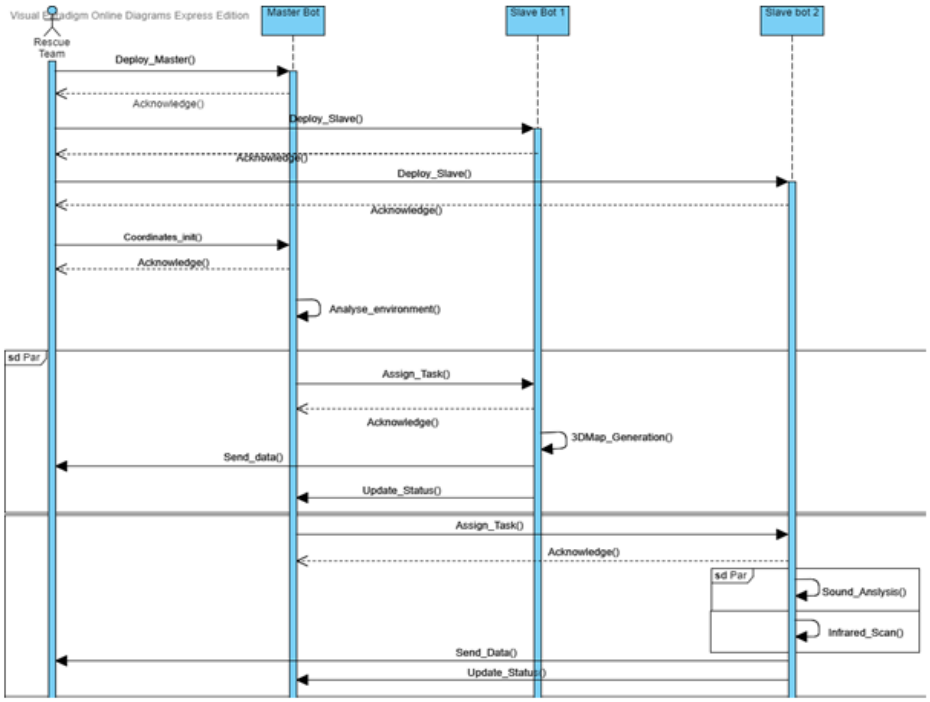

Figure 18: Sequence Diagram showing interaction between the actors when Survey Role and Detect Role is active

11.3 Real Time State Chart

The Real-Time State Chart is used to define the real time behaviour of the system. It is a combination of UML state machine diagram and timed automata. The overall system RTSC is developed and verified on *UPPAAL*. The state chart defined in the model corresponds to the roles mentioned in Subsection 11.1.

Figure 19 represents the state chart sequence of the Robot rescue assistance system. Initially the system is in activate state. *move_bot* signal is triggered which enables the Leader robot to start moving towards the rescue area. This is followed by signal to trigger the movement of the follower robots. Once all the robots are in motion, the begin signal is triggered to start the rescue operation. This is followed by requesting the *follower_info* and synchronizing all the robots as mentioned in state *sync_follower*. The *follower_info* is requested by the leader role. The coordinator role is responsible for establishing proper

coordination between the activation and deactivation of roles on a robot. After all the robots have been synchronized, signal is triggered to start the survey of the rescue area. This is carried out by survey role of the Leader robot. The Leader robot identifies the location of the area where the detection should take place. On receiving the position from the Leader robot, the Follower robot reaches the target and start searching for the victims. The find signal is triggered to ensure the detection of victims. Once the operation of detection is complete, the detect role of the Follower robot triggers the *mission_complete* signal. After receiving the *mission_complete* signal from both the follower robots the state moves to the initial activate state. The switching of roles can also be performed before moving to the activate state. This also ensures that there is only one robot acting as Leader robot at any instance of time.

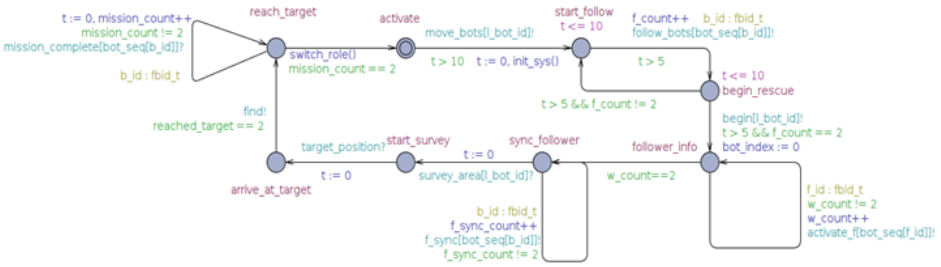

Figure 19: RTSC of Robot Rescue Assistance System

The Real-Time state chart of the coordinator role in demonstrated in Figure 20. It establishes the coordination between the leader role and follower role of the robot. A robot cannot be in a leader role and follower role at the same time. Initially the coordinator role is in ready state. It waits for the signal to be received from the state chart defined in fig 9. Based on the signal received, the coordinator activates the leader role or the follower role. Upon receiving the *survey__done* signal, the coordinator role of Leader robot moves to its initial state. Upon receiving the *detect__complete* signal, the coordinator role of the Follower robot triggers *mission_complete* signal and moves to initial state.

The Leader role is responsible for obtaining the position of all the Follower robots and analysing the health data of the Follower robots. The initial state of the Leader role is *in_motion*. After receiving the *begin* signal from the state chart defined in Figure 21, the leader role requests for the position and health of the follower robots. Once the health and position data are analysed, the leader activates the survey role of the robot. The survey role is tightly coupled with the leader role. This means that the Leader robot has an active leader role and survey role. Under exceptional cases, some tasks related to survey role for example, generation of 3D map can be assigned to the Follower robots. However, this part is not covered in the proposed system and is a part of future work.

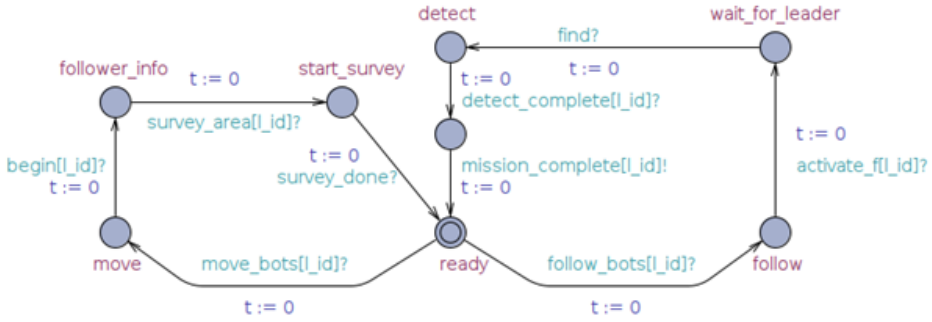

Figure 20: RTSC of Coordinator Role

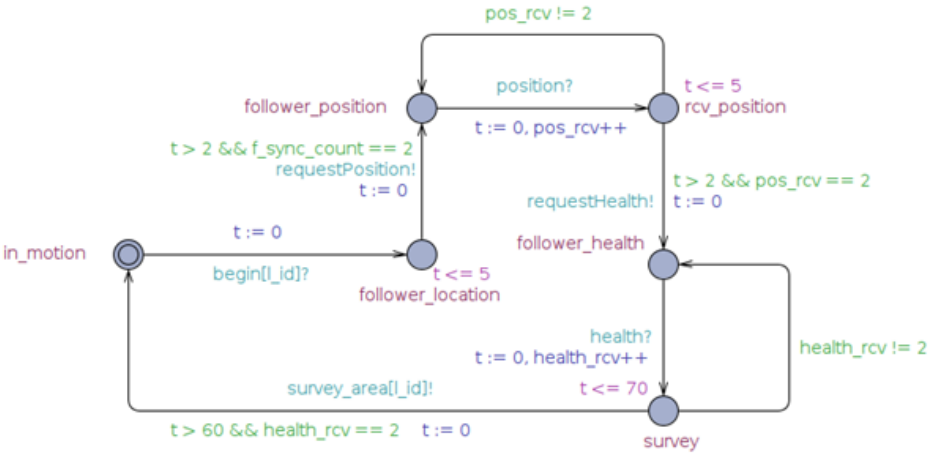

Figure 21: RTSC of Leader Role

The Follower role is the mirror opposite of the Leader role. The initial state is *in_motion*. After receiving the *f_sync* signal from the state chart defined in Figure 22, it moves to state *wait_for_master*. It provides the position and device health information to the Leader robot. On receiving the *target_position* signal from the Leader robot, it starts its movement towards the target. Upon reaching the target, it goes back to its initial state.

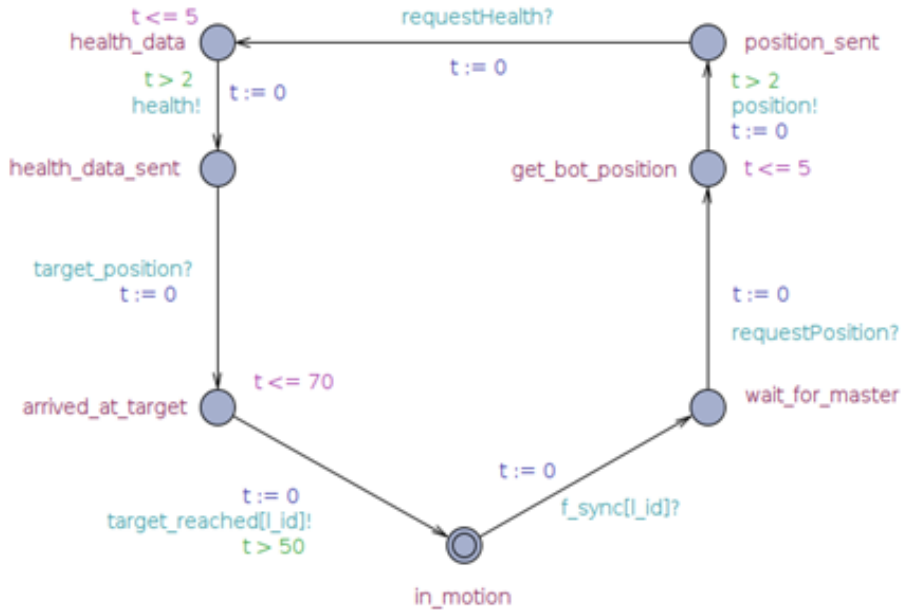

Figure 22: RTSC of Follower Role

Figure 23 demonstrates the behaviour of Survey role. The initial state of the survey role is *start_survey*. On receiving the signal to survey area, the survey role of the Leader robot is activated. It identifies the region for doing the search operation for the victims and notifies its location to the Follower robots.

The initial state of Detect role is *start_scan* as depicted in Figure 24. It waits for the Follower robot to reach to the target. As soon as the Follower robot arrives at the target, it starts the detection activity upon receiving the find signal from the Robot rescue assistance system state chart defined in Fig 9. Upon completion, it sends the signal *detect_complete*, which is received by the coordinator role (Figure 20).

The verification for the overall system is done in **UPPAAL** which is shown in Figure 25. The simulation results ensure that the overall rescue assistance system designed is deadlock free.

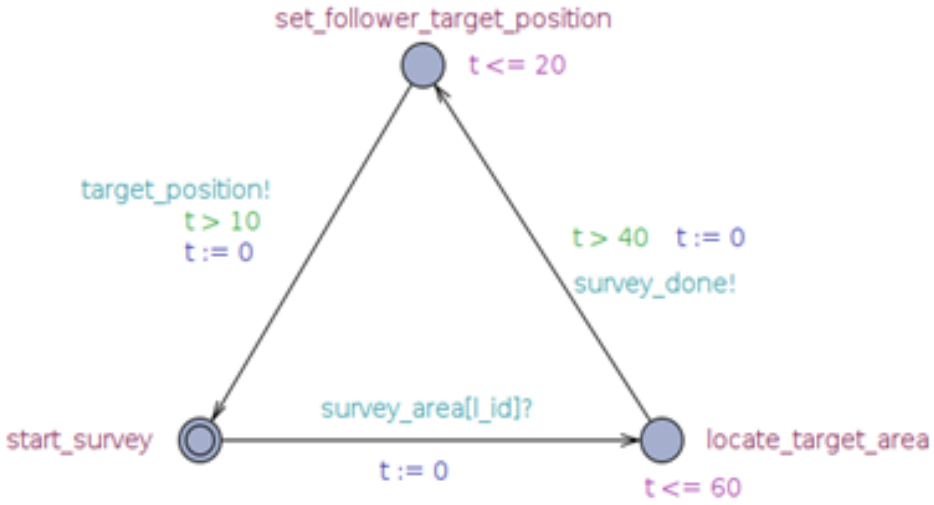

Figure 23: RTSC of Survey Role

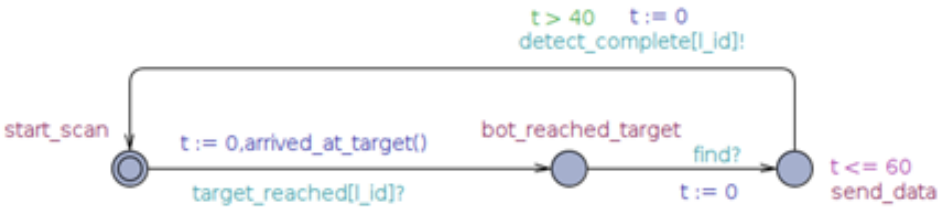

Figure 24: RTSC of Detect Role

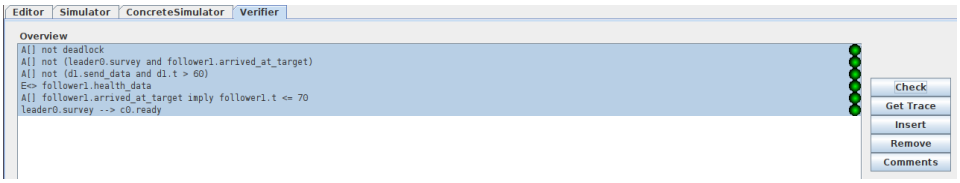

Figure 25: Test Cases for Verification

12 Cognitive Layer Decision Tree

The quality of data transmitted is the backbone of rescue operation. It is important, during run time, to enhance the quality of data being transmitted whenever possible. Performance improvement is also a part of Cognitive Layer as depicted in OCM Model in Sect. 10. Optimization loop is a procedure run by the master to optimize the communication parameters of the slaves. This change in communication parameters of the slaves changes the physical layer transmission properties of the slaves, which in turn improves the quality of transmitted data. The slave parameters used by the master for running the optimization loop are Signal Strength, Interference, Error Rate and Tx/Rx synchronization. Depending on the algorithm in the cognitive layer at the master, the decision whether to run the optimization loop is made depending on the input parameters.

As explained in the introduction we have chosen Signal Strength, Interference, Error Rate and Synchronization as our attributes. The features are “High”, “Medium” and “Low” for each of the first three attributes respectively, and “Synchronized” and “Out of Sync” for the Synchronization feature. Constant relay of data is imperative for the bot as other bots depend on it but more so because human lives are at stake.

The bot prioritizes clear communication and at any point of time if three or more of the attributes are in a bad condition then it runs the optimization loop to calibrate the various parameters to ensure a robust communication. For example, if the Signal Strength is low, Interference is high, Error rate is high and the bots are synchronized, we can see that three of the attributes are in a bad state and therefore, the bot makes a decision to run the optimization. Conversely, if the Signal Strength is high, Interference is low, Error rate is low and the bots are synchronized then there is no reason to run the optimization loop. Based on the this reasoning the initial dataset was made and the decision tree was plotted. **Gini index** was used to form the decision tree. It measures the impurity of a particular node. The equation for Gini index is given by[6]

$$1 - \sum_{i=1}^{c} {P_i}^2$$

The initial decision tree was not that complex and each of the layer was branching into one node which had a Gini index of 0. This is because the initial dataset was simple and had a lot of pure subsets. By pure subsets, it means that for a particular feature of an attribute there are all true answers. From the initial dataset and decision tree, when we checked the feature Error Rate, it was seen that whenever the error rate was “High”, optimization was always performed. This was a pure subset. We could see the same case for the feature Synchronization. Whenever it was “O” (out of sync) then optimization was performed. This

provided another pure subset. Hence Gini index was 0. So, more number of cases were added to the initial dataset which is shown in Figure 26. After more cases had been added to the dataset, it was found that there were lesser pure subsets available. Addition of various versatile cases to the dataset had made the dataset more complex. Using this dataset, the final decision tree was plotted which is shown in Figure 27.

Addition of more values to the dataset minimized the purity of subsets due to which there were a greater number of branches in the decision tree[5]. From Figure 27 it can be see how the algorithm splits the nodes. The entire dataset has been split in such a way that it uses 90% of the dataset to train the decision tree and the remaining 10% to test the decision tree, which is 26 cases for training and 3 cases for testing. The term "value" in each of the boxes signifies how the data has been split into "yes" and "no" branches at each node. The term "samples" depicts how many samples are being considered by the node.

13 Decision Tree Implementation Code

```
import pandas as pd
import os
df = pd.read_csv("Data_.csv")
lf = df

lf['Signal_Strength']=lf['Signal_Strength'].map({'Low':0,'Medium':1,
                                                 'High':2})
lf['Interference']= lf['Interference'].map({'Low':0,'Medium':1,'High':2})
lf['Error_Rate']= lf['Error_Rate'].map({'Low':0,'Medium':1,'High':2})
lf['Synchronization']= lf['Synchronization'].map({'O':0,'S':1})
lf['Optimization']= lf['Optimization'].map({'Yes':0,'No':1})

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.datasets import load_iris
from sklearn import tree
from matplotlib import pyplot as plt
import graphviz
from graphviz import Source

Feature_Columns=['Signal_Strength','Interference','Error_Rate',
                 'Synchronization']
X= lf[Feature_Columns]
Y= lf['Optimization']
```

```
X_Train, X_Test, Y_Train, Y_Test= train_test_split(X, Y, test_size=0.1)
print("Training Set")
print (X_Train)
print("Testing Set")
print (X_Test)
clf = DecisionTreeClassifier()
train_tree= clf.fit(X_Train,Y_Train)
fig=plt.figure(figsize=(24,24))
tree.plot_tree(clf, feature_names=Feature_Columns, filled=True)
fig.savefig('InitialAttempt.png')
#dot_data = tree.export_graphviz(clf, out_file=None,
                                 feature_names=Feature_Columns,filled=True)
#graph = graphviz.Source(dot_data)
#graph.render("Latest",view = True)
y_pred = clf.predict(X_Test)
print("Accuracy:",metrics.accuracy_score(Y_Test, y_pred))
```

The stepwise working of the code is written below.

- "Pandas" package was imported so that we can use it to read **CSV** (excel) file which contains the dataset.
- The decision tree could not be implemented if the dataset contained strings. After reading the values in the variable "df" the attributes were assigned numeral format which were as follows: "Low" = 1, "Medium" = 2, "High" = 3, "Synchronized" = 1, "Out of sync" = 0.
- All the packages required for plotting the decision tree was imported from sklearn[4].
- We split the dataset into training and testing where 90% was the training dataset and 10% was the testing dataset.
- After this, the training dataset was provided to the algorithm to plot the dataset.
- Once the dataset had been fed to the algorithm, the visualization of the decision tree was done using "matplotlib" package.

Signal_Strength	Interference	Error_Rate	Synchronization	Optimization
Low	Low	Low	S	No
Low	Low	High	O	Yes
Low	Medium	Low	S	No
Low	Medium	Medium	O	Yes
Low	High	High	O	Yes
Low	High	High	S	Yes
Low	High	Medium	S	Yes
Low	High	Low	S	No
Low	Medium	High	S	Yes
Medium	Low	Medium	S	No
Medium	Low	Low	O	Yes
Medium	Medium	High	S	Yes
Medium	Medium	High	O	Yes
Medium	Low	Low	S	No
Medium	High	Low	O	Yes
Medium	High	Low	S	No
Medium	Medium	Medium	S	No
Medium	High	High	O	Yes
Medium	High	Medium	S	Yes
High	Low	Low	O	No
High	Low	Low	S	No
High	Medium	Medium	S	No
High	Medium	High	S	No
High	High	High	O	Yes
High	High	Medium	S	No
High	Low	High	S	No
High	Medium	High	O	Yes
High	Medium	Low	O	No
High	Medium	Low	S	No

Figure 26: Final Dataset

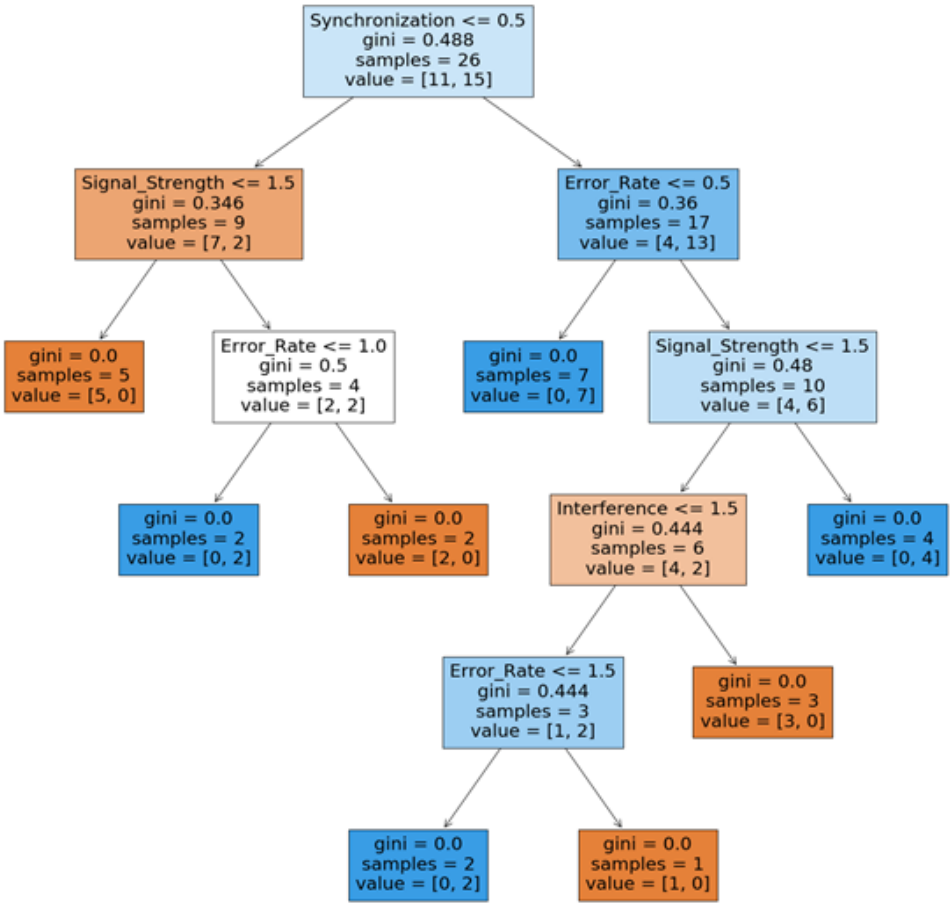

Figure 27: Final Decision Tree

Bibliography

[1] Gausemeier, Rammig, Schäfer, *Design Methodology for Intelligent Technical Systems*, Springer, 2014.

[2] Behrmann, David, Larsen, *A Tutorial on Uppaal 4.0*, Department of Computer Science, Aalborg University, Denmark, 2006.

[3] Martin Stigge, *Uppaal 4.0 : Small Tutorial*, Department of Computer Science, Aalborg University, Denmark, 2009.

[4] `https://scikit-learn.org`

[5] https://towardsdatascience.com

[6] https://blog.quantinsti.com/gini-index/

AFFIDAVIT

We (Adwait Bajpai, Anand Prakash, Ananya Reginal Frederick and Jitikantha Sarangi) herewith declare that we have composed the present paper and work ourself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance.