# Software design
# Team project – Deliverable 1

**Team number**: 20

**Team members:**

| **Name** | **Student Nr.** | **E-mail** |
|---|---|---|
| Ronan Hochart | 2614393 | rhochart@gmail.com |
| Mateo Vakili | 2619616 | mateovakili@gmail.com |
| Nick Kozanidis | 2616924 | nickkozanidis@gmail.com |
| Louk Onkenhout | 2619109 | loukonkenhout@hotmail.com |

## Table of Contents

**Author(s)**: <Ronan Hochart, Mateo Vakili>

## 1. Introduction

This paper will focus on the modeling and implementation of the ROBOSEARCH system. The system should be designed to be operated by anyone regardless of their knowledge and expertise regarding coding or more technical skills required to actually build the system. At its core the modeling will be used to layout how the system will be structured and how the bots will operate in and out of the system.

The main objective of our ROBOSEARCH system is to locate and log the location of mission targets in an environment. The mission targets are boxes identifiable by their color as well as size. They will be placed in a generated environment of max 25x25 meters with obstacles scattered around randomly. The whole environment is bordered by 4 large walls which act as a barrier for the environments area. Two different bots will be used in order to complete the mission objective which are both being controlled by the central station. The central station does most of the processing of the system, it communicates with each bot to make sure they are still functional and requests coordinates regularly in order to track mission progress. When a mission objective is found the coordinates are sent back to the central station which records and logs them in order to track not only the amount of objectives found but also their specific location. The central station is also responsible for initiating the mission and interpreting the mission parameters inputted by a human operator. This implementation allows for most of the processing implementation to be done in one class while the two types of bots remain quite simple only having to constantly send data to the central station to process. One responsibility not covered by the central station is actual bot navigation, the bots have their own navigation system, allowing them to operate in a semi-autonomous fashion. The AreaBot uses its bumper sensors to find the right direction it has to take after collisions. The SearchBots uses both sonar and bumper sensors to avoid collisions. The SearchBots do this by using a random coefficient to prevent the bots from getting stuck or choosing the same direction over and over when it fails to recover. The strategy used to avoid obstacles by the bots will be improved in future implementations by adding a navigation system. The navigation system will guide each bot based on its mission.

The first type of bot is named AreaBot. Between 1 and 3 of these bots will be used and their goal is to simply figure out the area of the environment. They will continually update the maximum coordinates while moving through the environment, and report back those maximum values found. After this task is complete the AreaBots are done with their part of the mission and simply deactivate.

The second type of bot is named SearchBot. Two to ten of these bots will be used depending on the area given by the Area Bots and their goal is traverse the environment and locate every mission objective they come across. Once the central station has determined that enough of the environment has been explored they will end their search. This is possible due to the central station knowing the area of the environment and therefore it is able to determine how much of the area has been covered by the searchbots. This is done by first splitting the area into an even grid and creating a matrix to represent the grid. The central station will then periodically receive the location data of each bot and marking that location as “discovered” in the matrix (this is done through simply flipping a 0 for undiscovered to 1 for discovered in the matrix). When a sufficient amount of the matrix and therefore environment has been marked as discovered the central station will terminate the mission and mark it as complete. Whenever a SearchBot finds a box it will first utilize color recognition to identify if the object is a mission target. If the object turns out to be a mission target it will send a message to the central unit with its coordinate location signaling that a target exists at this location. It will then proceed to contour the mission objective and send 3 other coordinates so that each side of the mission objective is logged.

# 2. Requirements Specification

**Author(s)**: <Louk Onkenhout, Nick Kozanidis, Mateo Vakili, Ronan Hochart>
2.1 Requirements

## 2.1 Requirements

**Functional requirements**

| # | Short Name | Description |
|---|---|---|
| F1 | Coverage of environment | **The system shall have good coverage of the environment space by description of what a location holds. (**the system has stored coordinates covering 95% of the environment, the coordinates are directly related to any objects at those coordinates**)** |
| F2 | Identification of the environment area | **The area bot shall send the computed area to the central system. (**the bot will calculate the area by finding the maximum x and z coordinates in either negative or positive direction. this is required since the environment's area is unknown**)** |
| F3 | Identification of all mission targets | **The identification bots shall send the surrounding coordinates of every located mission target to the central system. (**the bot will detect with obstacle is a target through identifying the color of the object and its geometric shape, it will then counter it saving and sending the list of coordinates that surround the target**)** |
| F4 | Color identification | **The identification bots shall be able to distinguish between different colors from a camera feed (**this will be done by taking a snapshot of what's in front of the bot when it encounters an object, it will then process and analyze the image and determine the color of the object**)** |
| F5 | Obstacle avoidance | **The bots shall be able to avoid obstacles without getting stuck (**whenever an obstacle is encountered the bot will first stop moving and then pivot either left or right and continue moving forward. This will prevent the bot from ever getting stuck as eventually it will redirect itself**)** |
| F6 | Allow for change or cancelation of mission | **The system shall be able to alter the properties of the mission at any given point in time.(**the color of the objects being searched can be altered**)** |
| F7 | Identification and logging of obstacles | **All obstacles encountered by the bots will be contoured and logged for identification.(** the coordinates will be recorded into a matrix with the determined type of the object**)** |
| F8 | System Navigation | **All bots shall have a navigation system** (The navigation system will get the next position in which the robot has to go using its current position and the mission of the bot. This will also ensures more coverage of the area in which the bot is assigned for) |

| | | |
|---|---|---|
| F9 | Central Station-bot communication | **All bots shall communicate only with the Central Station.**(the area bot will send information to central station regarding the environmental size while the search bots will await said information to initialize their search) |
| F10 | Authorization System | **Central Station shall have an authorization system. (** the central station will assign unique IDs to each bot that it deploys. Therefore it is able to confirm the access privileges to certain services provided by the Central Station) |

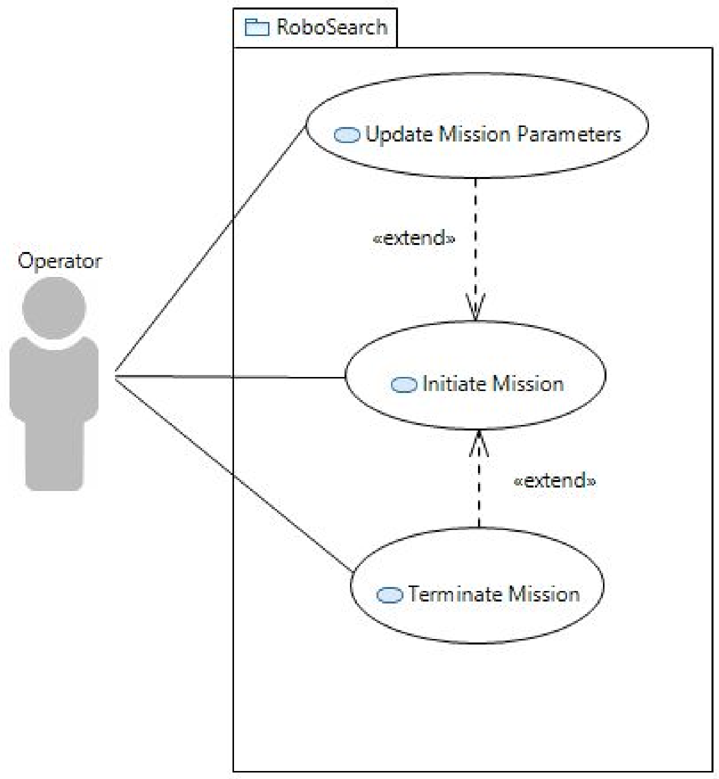
**Non-functional requirements**

| # | Short Name | Description & reasoning |
|---|---|---|
| NF1 | Response Time [Performance] | **Each rover shall react to the presence of an obstacle within 50 milliseconds.**(50 miliseconds provides enough time for the robot to not only try and avoid the obstacle but also come to a full stop if needed) |
| NF2 | Obstacle Avoidance [Safety] | **A rover shall always be at least 1 meter from obstacles, other robots, and human beings.** (1 meter would be enough space so that small errors in either human or robot movement/reaction times will not be enough to cause an accident) |
| NF3 | Swarm Size [Efficiency] | **The swarm size shall be adjusted in accordance with the environment area size**. (i.e a small number of bots will be instantiated for a small environment). |
| NF4 | Availability of System [Availability/Reliability] | **The system shall be consistently available to the user to operate.**(The system will be available to the operator unless it is disconnected ) |
| NF5 | Dependability of bots [Dependability] | **The bots have a dependency on the Central Station, but shall be autonomous to the extent of their own set of actions.** (this limits how dependent on the central station the robots are, therefore if the connection drops for a little bit or if the central station is busy communicating with another robot the robots don't stall out and can instead continue with their mission) |
| NF6 | Central management unit communication [Manageability] | **The bots shall constantly communicate with the Central Station to manage the behaviour of the bots in a more effective way.** (more information from the bots will help the central unit in its management process). |
| NF7 | Inter-bot Communication is Unavailable [Manageability] | **The bots  shall only communicate with the central system and not with one another**. (this streamlines the system and insures that all communication goes through the same hub avoiding possible bugs or issues that come with many robots communicating with many other robots) |
| NF8 | Communication acknowledgement [Safety/Reliability] | **There shall be confirmation of acquired information by all parties involved in the system.** (this feature will allow the central station and robots to make sure that any message sent has been received. Thus avoiding any missed logging if a message was simply corrupted on the transmission |
| NF9 | User-friendliness [Usability] | **The system shall be user-friendly, i.e. only easily understandable functions will be presented to the user.** (an operator and therefore user should be able to use the system and give it missions without having the technical background needed to hardcode the missions into the system) |

| | | |
|---|---|---|
| NF10 | Faulty bot self-termination [Reliability] | **A bot shall shut itself down on failure, if possible.** (if an error occurs and the bot is still operational, it will turn itself off) |
| NF11 | Faulty bot external termination [Reliability] | **A bot shall be terminated when unresponsive.**(if an error occurs and the bot cannot shut itself down, the central station will terminate the bot**)** |
| NF12 | Authorization System | **Central Station shall have an authorization system. (** the central station will assign unique IDs to each bot that it deploys. Therefore it is able to confirm the access privileges to certain services provided by the Central Station) |

## 2.2 Use Cases

Removed two use cases related to internal operations of the system and the Central Station actor.



**Use Case 1:**

| | |
|---|---|
| **Name** | Initiate Mission |
| **Short description** | The Operator initializes the ROBOSEARCH mission management system |
| **Precondition** | - There is no active mission<br>- All required parameters have been provided |
| **Postcondition** | - The ROBOSEARCH system is executing a mission plan with the provided parameters |
| **Error situations** | - Failed deployment of the mission<br>- Central station crashes during mission deployment |
| **System state in the event of an error** | - The ROBOSEARCH system is not executing a mission plan |

| | |
|---|---|
| **Actors** | - Operator |
| **Trigger** | - The Operator requests mission initiation |
| **Standard process** | 1. The Operator provides parameters to ROBOSEARCH<br>2. The Central Station initializes all needed bots<br>3. The Central Station sends out mission parameters to all bots that require it to operate<br>4. The Central Station reactivates the bots to perform their respective tasks |
| **Alternative processes** | 1. (See standard process 1)<br>2′. The Central Station cannot initialize enough bots, it will wait until enough bots are available<br>3-4. (See standard process 3 and 4) |

**Use Case 2:**

| | |
|---|---|
| **Name** | Update Mission Parameters |
| **Short description** | On operator request, the mission parameters of the active mission are altered. |
| **Precondition** | - There exists an active mission<br>- The new parameters are different from the current parameters and are valid |
| **Postcondition** | - The ROBOSEARCH system is no longer executing the initial active mission<br>- The initial bots have a different instruction set to execute |
| **Error situations** | - Failed deployment of the mission<br>- Central station crashes before the parameters have been updated throughout the system |
| **System state in the event of an error** | - The ROBOSEARCH system is still executing the same mission as before the central system attempted to send out a new command |
| **Actors** | - Operator |
| **Trigger** | - Operator requests for the mission parameters to be updated |
| **Standard process** | 1. Operator provides new parameters to ROBOSEARCH<br>2. Parameters are updated throughout the system<br>3. Central Station updates the bots instruction sets |
| **Alternative processes** | 2′. Central system determined the new parameters are either corrupted or a parameter is missing<br>3′. Ask the operator for a new set of parameters<br><br>3′′. Central station couldn't connect to the system<br>4′′. Central station will wait a set amount of time and try the connection again<br>5′′. Proceed from standard step 2 |

**Use Case 3:**

| | |
|---|---|
| **Name** | Terminate Mission |
| **Short description** | On operator request, the current mission is terminated |
| **Precondition** | - There exists an active mission |
| **Postcondition** | - The ROBOSEARCH system is terminated |
| **Error situations** | - The signal for termination did not reach the system |

|  |  |
|---|---|
| **System state in the event of an error** | - The system continues the mission |
| **Actors** | - Operator |
| **Trigger** | - Operator requests mission termination |
| **Standard process** | 1. Operator requests for the mission to be terminated<br>2. Central Station deactivates the bots<br>3. Central Station reports termination result |
| **Alternative processes** | 2'. Central station is unable to connect to bots<br>3'. Central station retries connection every minute until one can be established<br>4'. continue from step (2.) |

## 3. Implementation remarks

**Author(s)**: <Mateo Vakili, Ronan Hochart>

### AreaBot

AreaBot is responsible for finding the area of the environment. The Areabot has four modes,namely"GET_AREA","AVOIDING_OBSTACLE","DEACTIVATED","REPORT_MISSION_TARGET". The purpose of calculating the area of the environment is that in future implementation we will be changing the swarm size based on the environment area for efficiency reasons. Moreover, the area is not known in advance to evaluate the coverage of the environment by the bots.

**AreaBot.OPERATION_MODES.GET_AREA**: The AreaBot finds the area of the environment.

**AreaBot.OPERATION_MODES.AVOIDING_OBSTACLE**: The AreaBot's bumper sensors detect contact with an object and was able to identify it as an obstacle. This is a problem as it could prevent the bot from moving and most importantly accomplishing its mission. Therefore the AreaBot will try to recover from the collision in this mode. To be able to find the Area of this rectangular shaped environment we need the area bots to make contact with all four walls. The Areabot will be able to figure out if it has hit something with its 16 Bumper Sensors. The main downside here is that we won't be able to figure out whether the object the AreaBot has made contact with is an obstacle, objective or a wall. Therefore the bot will have a set amount of time to search the area as much as it can to find maxZPositive, maxXPositive, maxXNegative and maxZNegative points. Then using those values it will input them into an equation to find the width and height as shown below:

width = max( max( maxXPos * 2, |minXPos| * 2), width )

height = max( max( maxZPos * 2, |minZPos| * 2), height )

**AreaBot.OPERATION_MODES.REPORT_MISSION_TARGET**: The AreaBot uses the width and height values it calculated to find the area of the environment and reports it back to CentralUnit. At this point the AreaBot will deactivate itself.

**AreaBot.OPERATION_MODES.DEACTIVATED**: The Area But deactivates itself.

### CentralUnit

This class represents the Central Station in our implementation. The Central Station is responsible for initializing the mission parameters as well as coordinating the execution of the mission. The mission will consist of finding boxes with a predetermined color using the available bots. There are two types of bots at the stations disposal namely the AreaBot and the SearchBot. The bots will communicate back to the central unit constantly to check whether they have accomplished their objectives; moreover, giving them certain instructions to follow.

The Central Station first sends the Areabots. There are two Areabots used to increase the accuracy and reliability of the system in case of failure. sendAreaBot() is responsible for this part of the mission which is part of the CentralUnit and is called in its Constructor.

The AreaBot has to communicate back with the CentralUnit. It will do this by using the AreaReadyACK(UUID id, double area) static function in CentralUnit. To make sure that the robot is allowed to communicate with the CentralUnit using this function their UUID id will be checked (before the central Unit deploys any bot it will assigns it a unique ID to be able to identify it later when they communicate back with the CentralUnit). For the AreaBots deployed in the missions the ids are stored in DeployedAreaBotsIDs so the CentralUnit can check them in AreaReadyACK.

## SearchBot

The SearchBot is responsible for covering the area and recording coordinates of the mission targets. The avoidance strategy used for this bot for the moment is random . When the bumpers detect any collision the sonar sensors will be used. The quadrant measurements of the sensors on front, left and right will be calculated to take the appropriate action to avoid the obstacles.[1] SearchBot has three operation modes namely, "TAKE_IMAGE","DEACTIVATED" and "ACTIVATED".

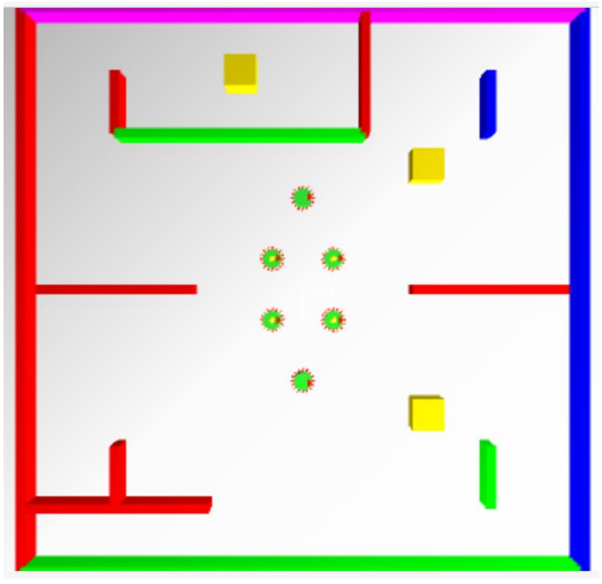
**SearchBot.OPERATION_MODES.ACTIVATED**: The SearchBot is activated by the CentralUnit to start searching.

**AreaBot.OPERATION_MODES.DEACTIVATED**: The SearchBot is Deactivated. the SearchBot will be in this mode when it is waiting for the AreaBot to calculate the area. When the CentralUnit gets the area back from the AreaBots, it will deploy the SearchBots to start their search for the mission targets by setting the SearchBots mode to SearchBot.OPERATION_MODES.ACTIVATED.

**AreaBot.OPERATION_MODES.TAKE_IMAGE**: The SearchBot will take pictures using it's camera sensor and extract the color of the object in which it has collided with. This will be used in future implementations to confirm the color of the mission target.

Note: As it is expected most of the implementation in AreaBot, Search Bot and Central Unit is not complete. the following will be done for future implementations:

- centralUnit can assign swarm size ( based on the area )
- searchBot's obstacle avoidance strategy will be improved by adding a navigation system to reduce the chance of failure.
- AreaBot's obstacle avoidance strategy will be improved.
- SearchBot will send coordinates back to central station. ( position of mission targets ) for this to happen Central station will assign a unique ID to each SearchBot. Also more functions will be implemented to make this communication possible.



## 4. References

[1] Reiners P. (2007, December 4). Retrieved from
https://www.ibm.com/developerworks/library/j-robots/