Distributed Indoor Mapping

Divya Ratnasami (w1v8)

James Park (e2d9)

Beichen Zhang (j2y8)

Howard Zhou (q6c9)

Haokun Chen (t1k0b)

1. Introduction and Background

The objective of this project is to build a distributed system in which the peers will work together to map out the area they are in. Each node will communicate its finding with its neighbours only when they come within a certain range of other robots. The robots will exchange their maps, resolve any conflicts between these maps and its local one, and coordinate how to explore the undiscovered areas. The robots will stop surveying the area when their energy level goes below a certain value, this will resemble real life situation because a robot would eventually stop.

2. High Level Overview

The aim of this project is to implement a distributed mapping system where Raspberry Pi robots (nodes) work together to map out a certain enclosed region. The topology will be time-dependent, that means robots will only be connected if they are within a certain radius of each other. Since the nodes are mobile, this would lead to multiple, different networks simultaneously. Each node will be mapping the area by following a path which will be an array of directions. Trying to emulate actual robots on a mission we introduce the notion of energy that will represent the battery level of the robot. This value will be represented numerically and vary inversely to the steps taken, that is the more steps a robot takes the lower the robots remaining energy will be. The robot will be initialized with an unique ID and its starting coordinate by the user. The user will use a consistent coordinate system therefore each robot will operate in the same coordinate system.

At the beginning of each journey it will start off with some initial energy, an empty map and no current path to explore. Each robot will generate a path for itself via the task allocation algorithm (explained in detail below) and start exploring this path. During its exploration, it will log each coordinate it traversed, the time it traversed, whether it is a wall or free space, and its current energy. Simultaneously, the robot will be broadcasting its IP and position. If during its journey it observes other robots it will follow the introduction protocol; that is, it will stop its motion, connect with these robots only if it is within its communication radius. Then it will send its map to them and also received the maps of its fellow robots.

Upon receiving the maps the robot will merge it's local map with each map given by its fellow robots. It will check if there are any conflicts and resolve them. For example, if Robot A recorded that there is a wall at coordinate (2, 7) but Robot B recorded that there wasn't a wall there, here the latest recording will take precedence as our assumption is that the environment is dynamic.

Upon map exchange each robot will compute the next tasks for all the robots via the task creation algorithm (explained in detail below). After this they will follow the task assignment protocol. From these newly computed tasks it will assign tasks for itself and its fellow robots. Each robot can either refuse or accept the task assigned to them. Only after they get a response, they will start to execute the assigned tasks. This eliminates the dependence of one node to assign tasks to the others in the network and is more robust to robot failures during this stage.

Upon task assignment the robots will continue exploring but now with this new task. Since the robots at this time will still be within the communication radius a time-communication threshold is introduced. This prevents communication between robots who just coordinated their task with each other so that they can start their new task without unnecessarily repeating the introduction protocol again. There will be a timer that will start following task assignment, and if the value of this timer is below the time-communication threshold, it means the robots cannot communicate even if they are within the communication radius.

Emulating a real-life situation the robots will eventually stop exploring when their energy goes below a certain value. However, if the user prefers to have the robot keep exploring, it can be configured in the software control by setting a very high initial energy or letting it be independent of steps.

A robot can fail at any point during its journey and our system will be equipped to deal with these. All of the robots current information will be logged onto disk, such as the robot's local map, path remaining to explore, coordinate, energy, and state. So if the robot re-connects it will be able to pick up its journey where it left off by visiting its log. If the log is empty it will know that it is just starting its journey.

3. Robot-Human Interfaces

The robots movement will be facilitated by a human operator. The robot will consist of a Raspberry Pi 3, 4 LED lights, 4 buttons, and a human operator. Each LED will represent a direction (North, South, East, West). Human will move based on the LED instruction. If the human can take a step to that direction, it will press button one to indicate that it is free-space and will press button two if it cannot take a step in that direction which indicates its a wall. The size of each step is consistent among human operators and will be represented in units in the Raspberry Pi rather than actual SI units. To improve the efficiency of the exploration, we have two more buttons representing the left and right bumper sensors on both side the robot. They are used to detect the presence of obstacles on the side of moving direction. For example, the right bumper button will be kept pressed when the robot is moving along a wall on its right and the robot is able to record the obstacles immediately to its side.

4. Assumptions/Invariants:

- The area will be finite, planar (2D) and objects inside may move around.
- Robot IDs, communication radius, and time communication threshold are set by the user.
- There are no malicious robots

5. Low Level Overview

Robot States

A robot can be in the exploring state, joining state, busy state, or waiting state. Figure 5-1 displays the robots state machine.

1. It is in the **exploring state** if there are no other robots within its communication radius. In this state, it is mobile and it will be completing its given path. It will simultaneously broadcast its IP address and position and listen to the IP addresses and positions from other robots. If it completes its task it will just recompute a new one and start exploring this new task.
2. A robot can be in the **joining state** if there are robots within the communication radius. The definition of "within the communication radius" is when the absolute distance between the robots are off by 1 unit. When they are within 1 unit, the robots will make connections. In this state, the robot will stop its motion (therefore its task). It will exchange its IPs with the robots in its proximity to form a network and confirm that all the neighbours within the network are consistent.
3. If the robot is in the **busy state**, it will follow the map merging protocol. When it is in this state, it will refuse connections from new robots that want to join it.
4. The other state a robot can be is in the **waiting state**, here the robot is within the communication radius of a robot in a busy state. Here it will still stop its motion and continue to poll the busy robot until the busy robot comes out of the busy state. When it comes out of the busy state, it will inform the robot in the waiting state. If the robot in the busy state dies, it will obviously not be able to respond back to the waiting robot so this robot will stop waiting and go into the roaming state.

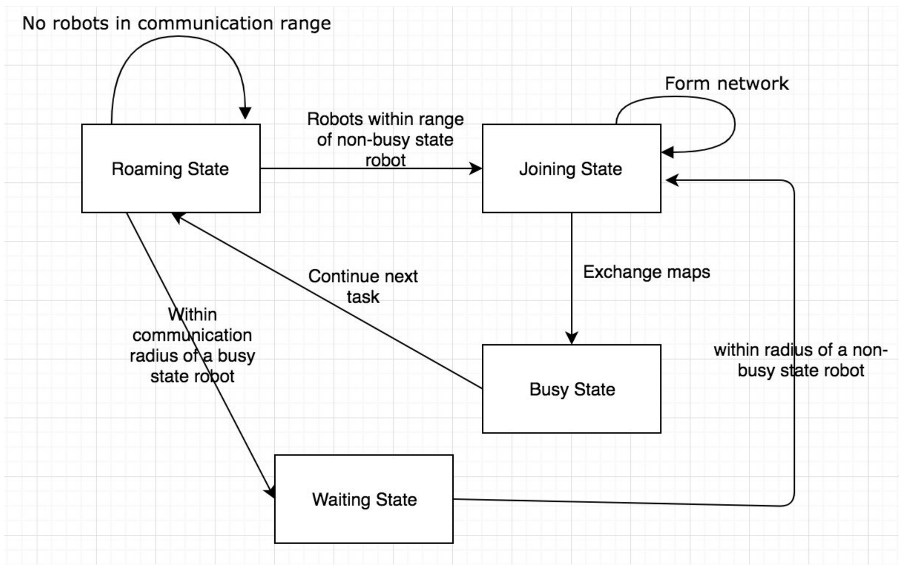
No robots in communication range
Form network
Robots within range of non-busy state robot
Roaming State
Joining State
Continue next task
Exchange maps
Within communication radius of a busy state robot
Busy State
within radius of a non-busy state robot
Waiting State

Figure 5-1: State Machine for Robot. Each robot initially will start from Roaming state and switch to other states based on the condition.

Inter-Robot Communication

We have investigated four different inter-robot communications namely NFC/Bluetooth/Wi-Fi Ad-hoc/Wifi . Based on current research, we would like to use Wi-Fi Ad-Hoc to both discover and communicate between nearby robots.Raspberry Pi will constantly broadcasting its static IP and exchange the locations with nearby robots. A “soft” threshold will be used to determine if two robots are close enough to start communication. As for connection to Azure, since Raspberry Pi cannot simultaneously be at ad-hoc and infrastructure mode, we need to either write a script to switch between two modes or introduce an access point with external connections. The final choice will be based on the reliability of individual solutions

Determine Robot Physical Coordinate

Due to the poor accuracy of civil GPS(4-7m), we propose to have all robots should start at the same point, the origin, and record down the direction and the number of rotations of their wheels to calculate its current location in reference to the the original point. However, such approach has cumulative errors due to mechanical inaccuracy. Therefore, we may need to add reference points such as a beacon or special object at the known location for robots to calibrate themself, if the accuracy of previous method is not satisfactory.

Protocols/Algorithms

Task Creation Algorithm

The goal is to cover the entire area evenly. This algorithm takes in the current map stored in the robot and generates N tasks for N robots in the neighborhood. The output will be identical among robots as long as the input is the same. As shown in Figure 5.1,the algorithm firstly calculates the center of current explored map by finding the extreme values of Xmax, Xmin, Ymax, Ymin of the explored region and modeled it as a rectangle envelope to calculate its center. The algorithm then picks N destination points with same radius spreading evenly on the 360 degree angular range. Then the algorithm estimates the distance from its location to the destination by calculating the sum of differences between in x and y coordinate, as if the robot will be traversing a " L " shape path which is the likely layout of a rectangle room / corridor. The final task generated includes the destination coordinate and estimated distances.

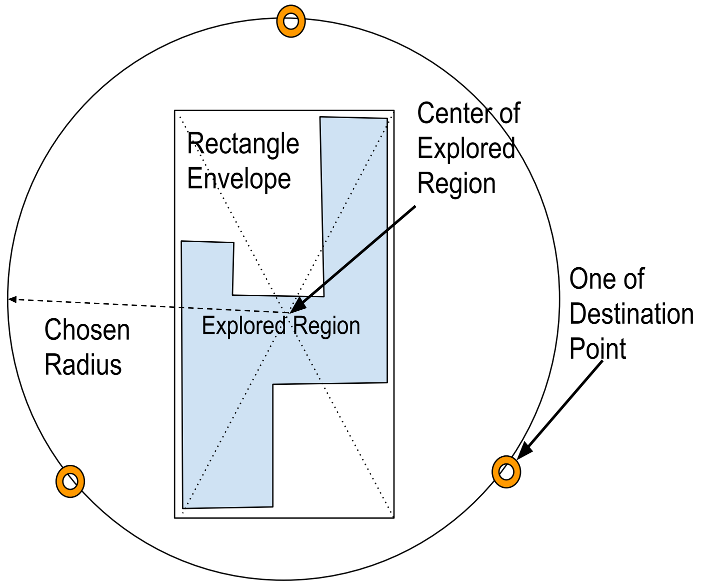

Figure 5.1: Task creation algorithms where the center of explored region is approximated by the center of its rectangle envelope and the destination coordinate is determined by the radius and angular location assigned by the algorithm as shown by orange dot .

Task Assignment Protocol

This protocol will take place after the robot calls the task creation algorithm. Here the robot will choose a task with the lowest energy for itself and choose tasks for the the other robots in the network at random from the tasks created via the task creation algorithm. Then the robot will send its fellow robots the task it created for them and its robot ID. The robot will wait for a response from its peers (whom are are alive), only when it receives all these responses will it continue to start exploring its new task.

A robot can choose to either accept or refuse a task that it is assigned by another robot. They will decide to accept the task assigned to them by the robot with the lowest ID and therefore refuse the tasks assigned by robots with higher IDs including itself.

The following time sequence diagram describes the basis of this protocol where they are all in the busy state.

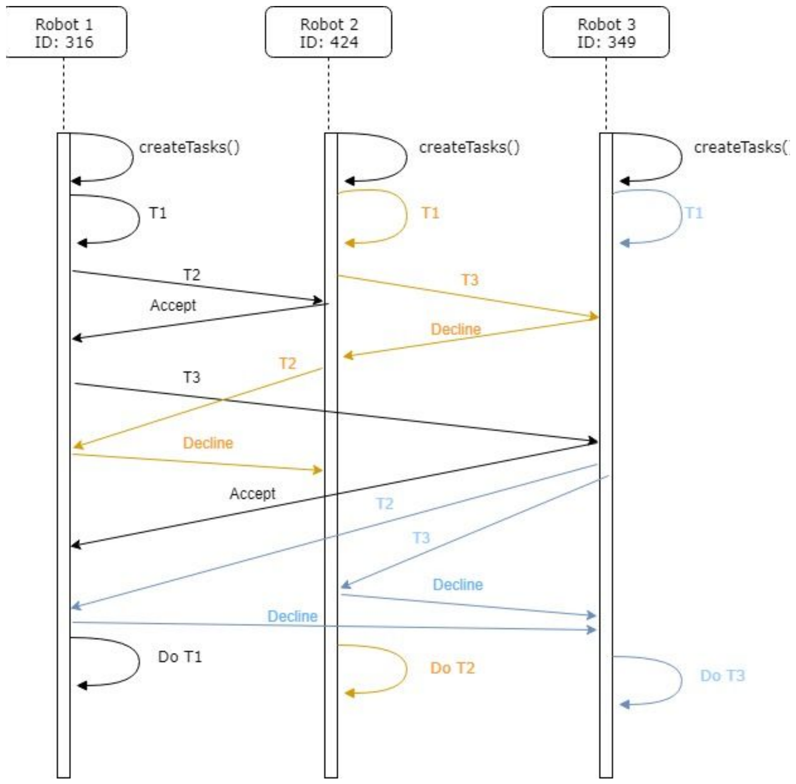

Figure 5-2:Time Sequence Diagram for Task assignment algorithm for a group of 3 robots in the same proximity.T1 is the path with the lowest energy and R1 has the lowest ID.

Map Merging

Our aim is to obtain an uniform map of the robots in proximity of each other. The robots will send a map payload which will consist of the Robots ID, map, and an end tag indicating the end of the package. A robot sends back an acknowledgment that it received the robots map if the payload contains an end tag. If a robot doesn't receive this acknowledgment within a certain time it will resend the map if the robot is still alive.

Upon exchanging map each robot will check for any conflicts between its current map and the one it received. A conflict happens when the same coordinate gets labels as an “obstacle” and “free space” by to different robots. To resolve conflict, each robot will take the lastest measurement as the correct one.

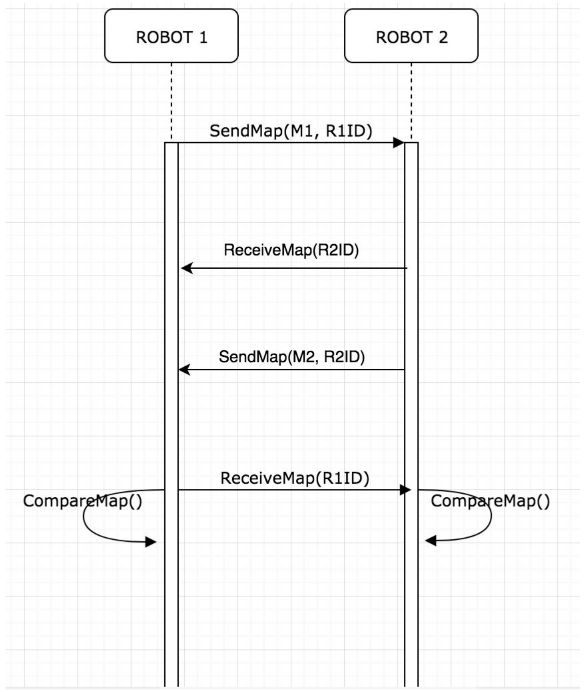

Figure 5-3:Map Merging Algorithm between two robots in the same proximity.

Handling Disconnections/Failures

Robot Failure during Busy state

The robot can fail a many points during this state. Each robot will be sending and receiving heartbeats from each robot in the network so if a robot doesn't receive a heartbeat it will assume the robot is dead. It will just proceed with all the protocol that take place in this state as if that robot isn't in the network. For example, if a robot dies during merging maps, the other robots will continue merging maps with the alive robots in the network and continue the task assignment protocol without sending a task to the robot that just died.

Robot Failure during exploration

In the case where a robot failed while executing a task, the distributed system as a whole is still able to explore the area which is supposed to be explored by the failed robot. Since the algorithm on other robots will consider those under-exploration area as unexplored area and still assign other robots to the location until it is mapped. The potential downside is that two robots may be assigned to the same location under different robot groups, which will result in wasting of resources. It is mitigated however by the fact that when these two robots meet, one of them will be sent to a new task.

In the case where a robot fails and then reconnects, it will read the log stored on its disk and resume its unfinished task. If no record is available, it will run the task creation algorithm with its current map.

Robot Recovery

For a robot to recover from a system crash it will consult its log file to resolve it's current position and current task it was completing prior to the crash. Then it will automatically go into the exploring state and resume its task.

Server Task

Currently, all raspberry pi robots are expected to connect to the server on Azure (as discussed in the Basic Inter-Robot Communication). While the robots are exploring the room, they will push their updated map to Azure. Then a user can make a GET request to the Azure server to view the updated map from each raspberry pi. The Azure server will not participate in exploring the map.

6. Timeline

****W-G : Whole Group , B: Beichen, D:Divya, H:Howard, J:James, M:Michael***

March 2nd	● Initial Project Proposal **[W-G]**
March 5th	● Figuring out communication protocols **[B]**
March 9th	● Final Project Proposal **[W-G]**
March 10-11th	● Go to Lee Electronics and obtain 3 additional Raspberry Pi and extra SD cards. **[W-G]** ● Install raspbian os in each Pi and install the latest Golang **[W-G]** ● Set up the communication between robots using ad-hoc **[W-G]**
March 12th	● Start implementing Map Merging Protocol **[D, H, M]** ● Start integrating physical LED and buttons. Start learning to use go-rpio library **[J,B]**
March 13th	● Start implementing Task Creation Protocol **[W-G]** ● Keep working on Map Merging Protocol **[J, B]**
March 13-17	● Start implementing robot states. If task assignment protocol or merge assignment protocol are not done, then we will emulate some of the functions **[J, B]** ● Finish on Map Merging Protocol **[D, J]**

	● Resume working on Task Creation Protocol **[W-G]** ● Start working on Task Assignment Protocol **[D, H, J]**
March 18th	● Start integrating Azure to display **[W-G]** ● Start integrating GoVector and Shiviz into the project **[H, M, J]** ● Finish implementing state machine for robot **[H, M]** ● Finish implementing Task Assignment Protocol **[D, H]** ● Investigate and Prototype Physical Robots**[B, J]**
March 19th	● Try to finish integrating GoVector and Shiviz **[H, M]** ● Finish working on Task Creation Protocol **[D, M]** ● Have the Raspberry Pi output directions **[B, J]**
March 20th	● Start integrating Dinv into our project **[M]**
March 21-25th	● Finish up all low level protocols **[W-G]**
March 28th	● Test whole system **[W-G]**
March 31st	● Finish debugging **[W-G]**
April 4th	● Prepare for demo **[W-G]**

7. SWOT Analysis

Strengths:

- Beichen Zheng is an Engineering Physics student who won the first place in ENPH 253 (a robotics course). On top of that, he knows a lot about Raspberry Pi.
- All member are passionate about learning distributed system.
- All team members are persistent and never give up.
- All team member love Golang.

Weaknesses:

- Some team member have no prior experience with Raspberry Pi.
- No one on our team knows how to use Govector, Shiviz,Dinv.

Opportunities:

- Raspberry Pi has a very active community. If there is any roadblock on the hardware, it would be easy to get help online.
- Raspberry Pi has a very intuitive interface and a smooth learning curve.

- TAs and instructor are very helpful, we can get help online and offline.
- All knowledge acquired from lecture can be applied to our project.
- Our project does not rely on any infrastructure

Threats:

- All team member have heavy course load, therefore scheduling meetings could be difficult at times.
- Hardware integration has lots of uncertainties and technical limitations.
- One of the team members, Beichen Zheng, is considering of dropping the course. If that happens, a meeting with the instructor must be conducted to adjust the workload.
- Since this project requires additional hardwares, each team member needs to contribute at least $100 to purchase the necessary hardwares. At most, we will spend another $100 in case we need additional hardware.

8. Extra Features

Fully automated robot with wheels

If the time permits, we will build 3-5 robot cars equipped with battery, motor, sensors, Wi-Fi/bluetooth modules and signal processing chips to carry out the exploration tasks. The interaction with physical devices will be structured into an API such that the software could be easily switched from Human-robots to fully automated robots with minimum changes.

Gas/Energy Station

As a robot explored this new land it will be expending goes about its exploration it will use up its energy. To allow the robot to continue exploring while keeping this real life feature we can introduce a designated area that will allow for a robot to refuel its energy and continue on its journey.

Robot with a Goal

If time permits, we may give robots an ultimate goal to achieve such as finding a specific object in the room with OpenCV and a camera, while mapping the path simultaneously.