Distributed Mapping

Divya Ratnasami (w1v8)

James Park (e2d9)

Beichen Zheng (j2y8)

Howard Zhou (q6c9)

Haokun Chen (t1k0b)

Introduction and Background

The objective of this project is to build distributed system in which the peers will work together to map out the area they are in. Each node in the network is a mobile robot that will be surveying the area it is placed in. It will follow a gossip protocol communicate its findings with other nodes, that is only when they come within a certain range of the robot. The robots will exchange their maps, resolve any conflicts they view with the maps they received and its local one, and coordinate with each other to explore undiscovered areas. The robots will stop surveying the area when their battery power goes below a certain value, this will resemble real life situation because a robot would eventually to get its power source recharged.

High Level Overview

The aim of this project is to implement a distributed system where the nodes work together to map out the area they are placed in. The topology will be time-dependent, that is a nodes will only be connected to each other if they are within a certain range of each other, as the nodes are mobile this would lead to a different network as time progresses. This will not include a centralized server. Each node will be mapping the area by following a path which will be an array of directions. Upon start up each node will be assigned an ID by the user and start off with a empty map. It will generate a path for itself via the task allocation algorithm and start exploring this path. During its exploration, it will log each coordinate it traversed, the time it traversed, and whether it is a wall or free space. Simultaneously, the robot will be broadcasting its IP and position. If during its journey it comes into contact with other robots it will stop its motion, connect with these robots only if it is within its communication radius and will broadcast it's map to them and the other robots will do the same. Robots will only be able to communicate with each other only if they are within a certain radius (communication radius) of each other and if they haven't previously connected with each other within a certain time threshold, this will be further explained below and these limits will be set by the user.

Upon receiving the maps the robot will merge it's local map with each map given by it's fellow robots. It will check if there are any conflicts, that is 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. Since each robot follows this procedure in theory they should have consistent maps.

Upon map exchange each robot will compute the next tasks for all the robots via the task creation algorithm. Following this they will do the task assignment algorithm. From these newly computed tasks they will randomly assign each robot including itself a task and the corresponding task to its fellow robots. Each robot can either refuse or accept the the task assigned to them. Only after they get a response and task assigned to them by each robot can they actually do the task they decided to do. This eliminates depending on one node to assign tasks to the others network and is more robust to robot failures during this stage.

Upon task assignment they will continue exploring but with this new task. Since the robots at this time will still be within the communication radius but they just connected, we

introduced the a time-communication threshold which states which is the and repeat this procedure if they bump into other robots.

Emulating a real-life situation the robots will eventually stop exploring when their energy goes below a certain value, in this case electrical energy. However, if the user prefers to have the robot keep exploring, it can be configured in the software control.

A robot can fail at any point during its journey and our system will be equipped to deal with these. All the information of the robots local map, its current task, current coordinate, and state will be stored on disk so if the robot re-connects it will be able to pick up it journey where it left off.

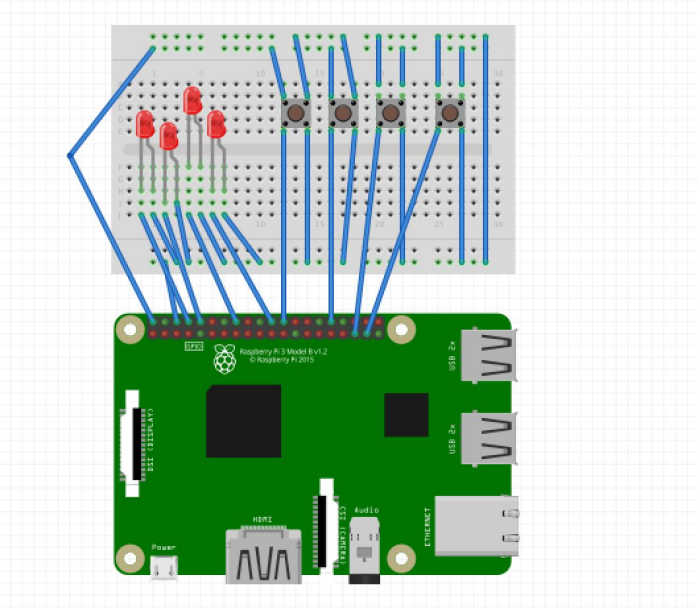
When the robot starts its journey it will be initialized with its starting coordinate so that all robots share the same coordinate system.

Robot's Movement

To simplify this project, each robot will not use motors and wheels to explore the room. Instead each robot will be carried by a human. In the diagram below (which was constructed using Fritzing), there is a Raspberry Pi 3 and a breadboard with 4 red led lights and 4 switches. The led lights will tell its human where to move (North, East, West, or South). If the human can take a step to that direction, it will press button one to indicate that direction is valid. If the direction is not valid, then button two will be pressed.

To maintain consistency, each robot's human will try to take the same size of step when the robot orders the human to move.

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.



Assumption:

These are the following assumptions that we have had:

- The robots will explore a finite and confined room. This includes the following
    - The room won't expand
    - Objects are allowed to move around but it will take some time for the change to be updated.
    - The area is planar
- The IDs of all the robots will be natural number and their IDs will be displayed using led lights. Each of robot's id will be unique such as the hardware series number or the MAC address of its wifi switch.
- After some of the robots finished the merging their map and allocating the tasks, each of these robots will ignore communication for 10 seconds to ensure they could have enough time to spread out.
- Communication radius and time communication threshold will be set by the user
- There are no malicious robots (ie. communicate even though they are not with in the communication radius etc.

Low Level Overview

States

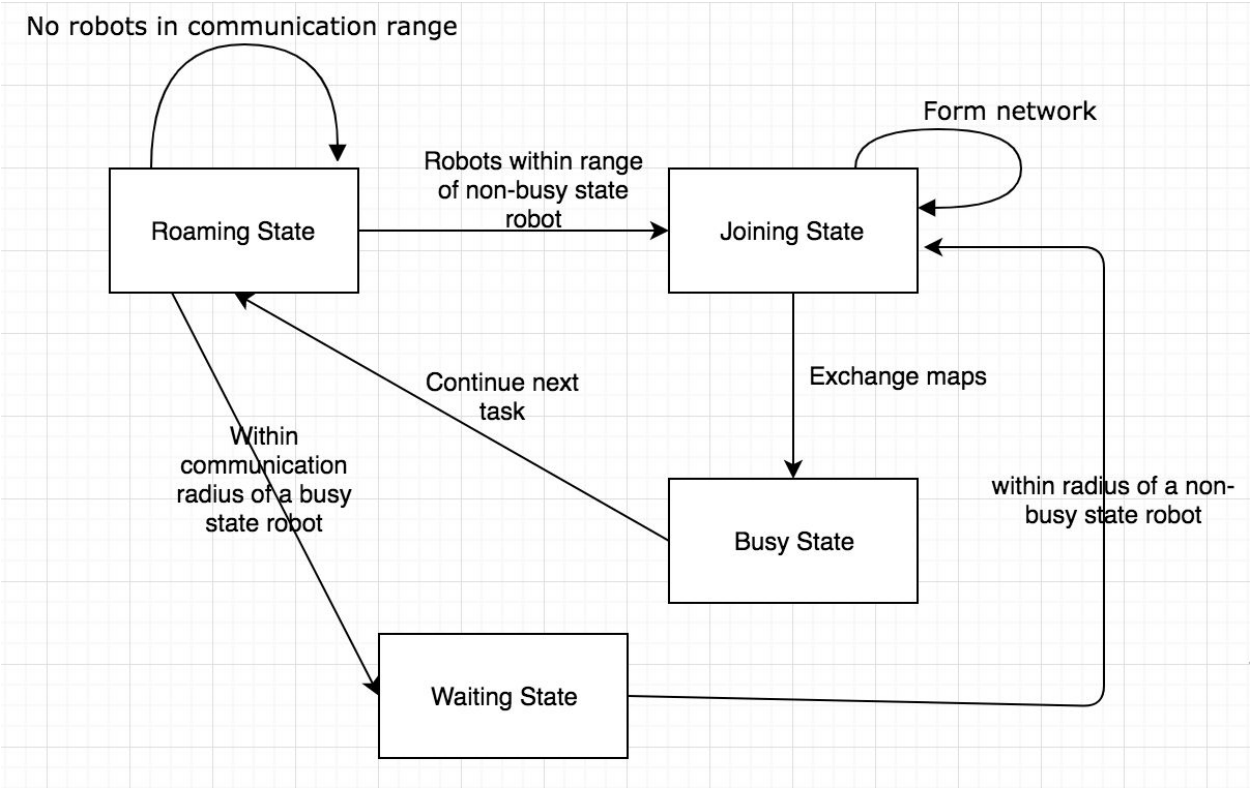
Robot States:

The robot can be in multiple states and this influences the actions it can perform.

A robot can be in the roaming state, joining state, busy state, or waiting state.

1. It is in the **roaming 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. While it is in roaming state, each robot will broadcast its IP address and position and listen to the IP addresses and positions from other robots. The robots won't take any actions with incoming if the other robots are within the communication radius.
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 it's 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, that is the robot in the busy state will return it's status to 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.

The diagram below summarizes the states.



Protocols

Basic Inter-Robot Communication:

We have explored four different inter-robot communications namely NFC/Bluetooth/Wifi Ad-hoc/Wifi and listed the pros and cons in the table below. Based on current research, we would like to use WIFI 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 determine if two robots are close enough to start communication. Although our priority is to configure Ad-Hoc network, as it is still uncertain if Raspberry Pi’s wifi module is capable to connect to Azure while configured as Ad Hoc mode, we may need to fall back to Bluetooth for inter-robot communication and Wifi(with external connections) .

	Pros	Cons	Planned Usage
NFC	Limit the communication range by design	-Short Range(4-cm) -Slow (1 kbytes) -Easily interrupted when multiple devices are in proximity	No long considered
BlueTooth	-Better range and speed than NFC	- Master-Slave pairing limits communication between two devices	Broadcasting the location and IP of the robot to its proximity.

		only per time. -Required to implement Go-Wrapper to existing Python library otherwise majority of the program will be in Python	
Wifi	-flexibility, speed, access to TCP/IP stacks and other golang library	-Dependence on WIFI infrastructure	Once IPs are exchanged through bluetooth, robots will use wifi to merge maps and collaborate on task assignment.
Ad Hoc Wifi	-Independent from WIFI Infrastructure	-Need to wrap C firmware library -Static IP Address Required -Depends on topology, an access point may need to be configured	Alternative to Wifi

Distance Measurement

As the indoor distance measurement hardware are rather expensive and difficult to integrate in nature, we would use “soft” distance measurement for robots by broadcasting their coordinates to each other. Each receiving robots will calculate the relative distance using its own coordinates. The downside is that this approach highly depends on the accuracy of the coordinate systems. If the accuracy is not satisfactory, we will use the signal strength of bluetooth as the complementary measurement to enhance the accuracy.

Determine Robot Physical Coordinate

Due to the poor accuracy of civil GPS(4-7m), we need alternative ways to determine the physical location of the robots. One proposed method is that all robots should start at the same point, the origin, and record down the number of rotations of their wheels to calculate its current location in reference to the the original point. However, such approach has significant cumulative errors due to mechanical inaccuracy. Therefore, we propose that if it is allowed to put some reference points such as a beacon or special object at the known location of the room to be explored. Whenever the robot pass through the reference point, it will recalibrate its coordinate system based on the beacon’s location. Whether we need the reference point or other technologies will depend on the performance of the prototype machine.

Task Creation Algorithm

This algorithm will create paths for the robots in the network. The inputs will be the current map (M) and the number of robots in the network (N), given this it will come up with N tasks one for each robot in the network. This task will be a path that the robot will explore and will be in an array of directions, for example [“west”, “south”, “west”, “east”]. Based on the current map, the algorithm will figure out the coordinates that haven't been traversed and create a path that will result in the unexplored area. Another constraint will be to create paths such that they don’t intersect.

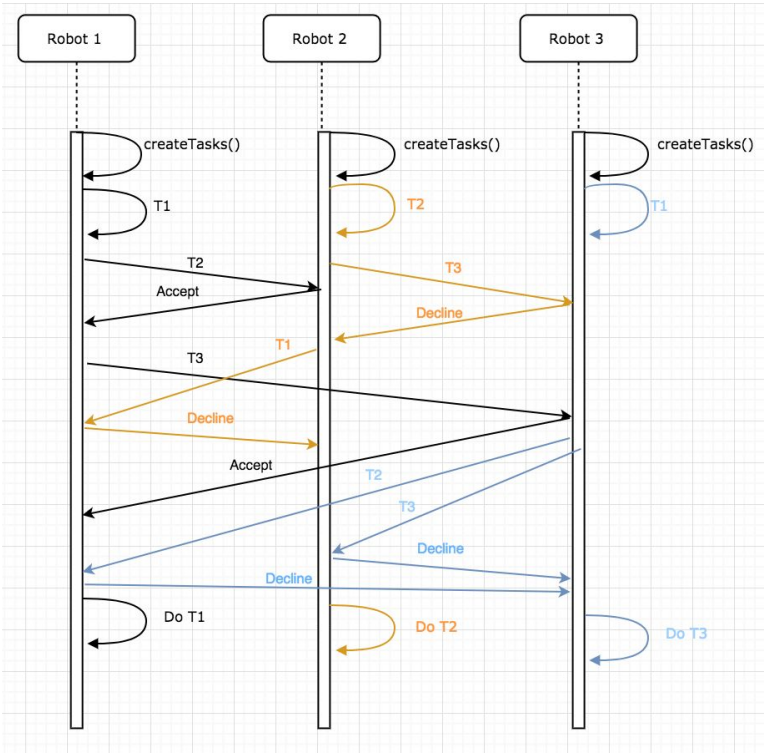
Task Assignment Protocol

This protocol will take place after the robot calls the task creation algorithm. Here the robot will choose a task for itself and 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 ID. The robot will wait for a response from its peers, only when it receives all these responses will it continue to start its new task.

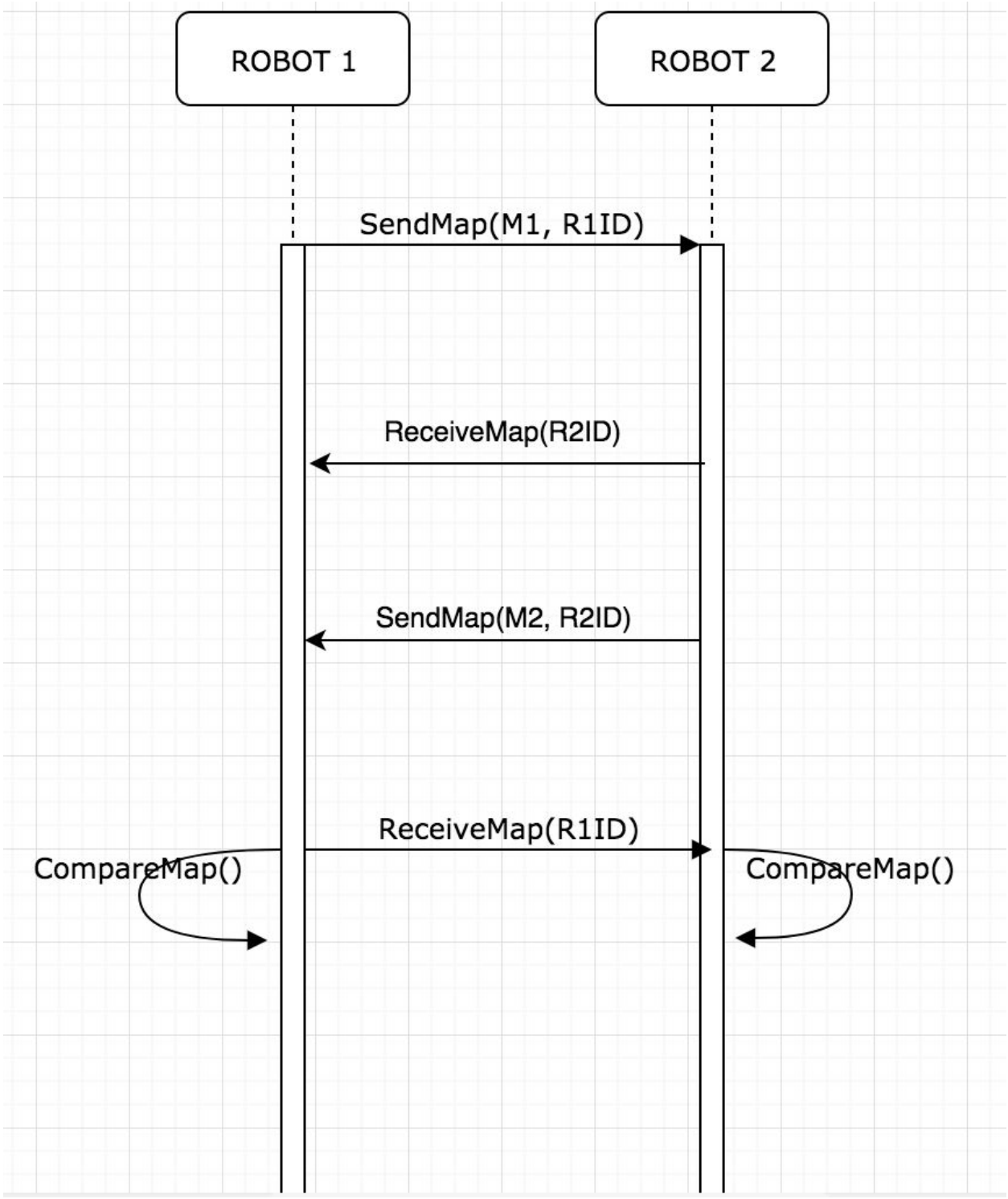
A robot can choose to either accept or refuse a task that it is assigned by another robot.

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.

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



Map merging procedure



Robustness of Task Allocation Protocol

In the case where a robot failed while executing an exploration task, the system is still able to explore the area which is supposed to be explored by the failed robot, because the algorithm to find available tasks 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 resulting 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 failed and then reconnected, it will read the log stored on its disk and resume last unfinished task. If no record is available, it will run the find-available-tasks function and pick the first one in the result.

Map Merging

Our aim is to obtain an uniform and most updated map for every robot in the proximity. The robots will broadcast a map package which will consist of the Robots ID, the robots map, and a end tag which as its name suggest will be placed at the end of the package. A robot sends back an acknowledgment that it received the robots map if it received end tag. Only when the robot which sent it's the to the network. Therefore, they will broadcast their own map to each other and merge the newly received maps with the current map in their memory. If conflicts arises, keep the mapping with the latest timestamp assuming each obstacles/wall on the mappings is timestamped. In this project, we will assume that obstacles positions are fixed (chairs don't move, doors don't close).

Map Merging Conflict Resolving Protocol

Upon exchange each robot will check for any conflicts between its current map and the one it received. A conflict is defined as a coordinate (spot) being labeled as a wall or free space. To resolve this, each robot will take the lastest measurement as the right one.

Stop Condition

At the beginning of the program, each robot will be assigned a certain amount of energy, when the energy level dips under a certain threshold, the robot will then stop roaming.

Handling Dead Robot

If one robot disconnects while exchanging information with the group in the proximity, rest of the robots will restart the map exchange and merging. The disconnected robot itself will revert its map to the last successfully merged version and start exploring on its own.

New / Revived Robot

If new robots joins the network or recovered from any form of system crash, it will firstly try to locate on the file system to see if there are any log files about its last interrupted exploration(Null for a new robot) and resume its exploration based on the log file.

Server Task

Currently, all raspberry pi robots are expected to connect to the server(on Azure) with WIFI and constantly upload their current maps. Server will be only responsible for displaying robots' local map and will not facilitate any communication to or between robots .

Timeline

March 2nd	● Initial Project Proposal ○ Accomplished by the whole group
March 5th	● Figuring out communication protocols ○ Accomplished by Beichen
March 9th	● Final Project Proposal ○ Accomplished by the whole group
March 10-11th	● Go to Lee Electronics and obtain 3 additional Raspberry Pi and extra SD cards. ○ Accomplished James, Haokun, Howard, Divya ● Install raspbian os in each Pi and install the latest Golang ○ Accomplished by James, Haokun, Howard, Diyva ● Set up the communication between robots using ad-hoc ○ Accomplished by the whole group
March 12th	● Start implementing Map Merging Protocol ○ Accomplished by Howard, Michael, Divya, Haokun, James ● Start integrating physical LED and buttons. Start learning to use go-rpio library ○ Accomplished by James and Beichen
March 13th	● Start implementing Task Creation Protocol ○ Accomplished by the whole group ● Keep working on Map Merging Protocol ○ Accomplished by at least two members
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 ○ Accomplished by James ● Finish on Map Merging Protocol (hopefully on March 14)

	○ Accomplished by at least two members ● Resume working on Task Creation Protocol ○ Accomplished by at least two members ● Start working on Task Assignment Protocol ○ Accomplished by one member
March 18th	● Start integrating Azure to display ○ Accomplished by the whole group ● Start integrating GoVector and Shiviz into the project ○ Accomplished by Howard, Michael ● Finish implementing state machine for robot (independent of the protocols) ○ Accomplished at least one member ● Finish implementing Task Assignment Protocol ○ Accomplished by one member
March 19th	● Try to finish integrating GoVector and Shiviz ○ Accomplished by at least two members ● Finish working on Task Creation Protocol ○ Accomplished by at least two members ● Have the Raspberry Pi change direction (i.e display the change using LEDs) when the buttons are pressed ○ Accomplished by one member
March 20th	● Start integrating Dinv into our project ○ Accomplished by Howard, Michael
March 21-25th	● Finish up all low level protocols ○ Accomplished by the whole team
March 28th	● Test whole system ○ Accomplished by the whole team
March 31st	● Finish debugging ○ Accomplished by the whole team
April 4th	● Prepare for demo ○ Accomplished by the whole team

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 Shiviz, getting extra credits could take some effort.

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 meeting time could be difficult at times.
- Putting hardware together is proven challenging and it could take a lot of time to get started.
- 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.

Extra Features

Fully automated robot with wheels:

If the time permits, we will build 3-5 robot cars equipped with battery, motor, sensors, wifi/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

When a robot roams around the map it would spend its gas/battery level. There will be designated area that will replenish a robot's battery level to some extent enabling the robot to roam longer.

Robot's 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.

Terminology

1.**Task**: an exploration mission for a single robot to execute. It will probably in the format of "explore and find a path from current location to physical coordinate (100,200)".
2.**Robot Group:** a temporary group of robot that is within a predefined physical distance and is engaged in inter-robot communication.