Eyantra Robotics competition task 4c

To start the client robots,
1.Run the following command in the working directory of the client robots.

```
iex -S mix
```

2. Now run the main function form the main module

```
//robotA terminal
>Task4CClientRobotA.main
//robotB terminal
>Task4CClientRobotB.main
```

This action will establish connection between the clients and the server so that messages and requests can be sent and received.
The main function will receive the start position in case of boththe robots and the goal location that need to be covered.
Now, once the start positions and the goal locations are received, the start function is called passing in the start position of the robot, once the robot is placed the status is sent to the server using the send_robot_status/[write airity] function and then the go_to_goal function is called which with the help of various helper function does travels to all the goal locations that has been assigned to the robot.

Whenever a robot takes any action like move, right turn, left turn. The actions are sent to the server through a topic named robot:status and the actions are updated on the LiveView through the robot:update topic.

The procedure to receive start and the stop position is as follows:

First the main function sends the request to the server to receive the start positions if the start position has not been updated on the liveview by the user a reply stating that the start position has not been updated is sent to the client and if the client receives such reply on the request then it sends the request again. This action is performed until the clients receive the start positions.

The same procedure is followed when the goal locations are to be received.

Once the goal positions are received the stop function is called recursively until all the goal locations are covered.

Flow of the program[to add more]

1. the Plantposition.csv file is read on the server and is parsed to find the goal locations that are to be covered.
2. Now the goals are divided among the robots such that the robot closest to the goal location will cover that goal location.
3. The decided goal locations are sent to respective client robots
4. The clients then call the stop/[write airity] function recursively to cover all the goal locations.

To Run the project:

1.Starting the server:-
Open a terminal in the phoenix app and type the following command

mix phx.server

2.Starting the clients:-
Open a terminal in the working directory of the clients and run the iex by the command

iex -S mix

Now Run the main function from the respective module by the following command in the respective terminal of the clients.

Task4CClientRobotA.main
Task4CClientRobotB.main

The main/0 function does the following jobs-

- Gives a starting point for the task for the client robots
- Receives Start positions from the server.
- Receives the goal location that needs to be visited by the respective robots.
- Calls respective functions from the module to perform the task at hand.

Procedure followed to receive the start position and goal locations from the ArenaLive.

1. When the start position of the respective robots are entered in the ArenaLive, It triggers the handle_event/3 callback with “start_clock” event.
2. The start positions are received in this callback and the start position parameters are updated via the assign function. The start position data of both the robots are broadcasted to a topic robot:get_position with event name of “startPos”
3. The start position data is routed to the RobotChannel module in robot_channel.ex file, Where it is received in the handl_info callback function.
4. The received start positions are updated in the socket by assigning functions so that when the client requests for the start position it can be accessed from the socket.
5. To receive the start position stored on the server the client sends a synchronous message on the topic robot:position to receive the start position in return. If the start positions are not updated on the server it receives a message stating that robot positions are not updated on the server. On receiving such messages the client keeps sending synchronous messages until the start position data is updated on the server and the data is received in the main function of the client.

6. In the same handle_event/3 callback function respective helper functions are called to do the following:
    - make_goal_loc/0 function is called to parse the data in the Plant_Positions.csv file to obtain the goal locations that need to be visited by the robots.

    - goal_distribution/6 function is called to distribute the goals to the robots.
    - Now, in this callback function the parameters of goal location defined via the assign function are updated.And the distributed goal locations are updated in the socket of the RobotChannel same way as the start positions.

7. To receive the goal locations from the server in the main function of the respective clients the same procedure as receiving the start position is followed.

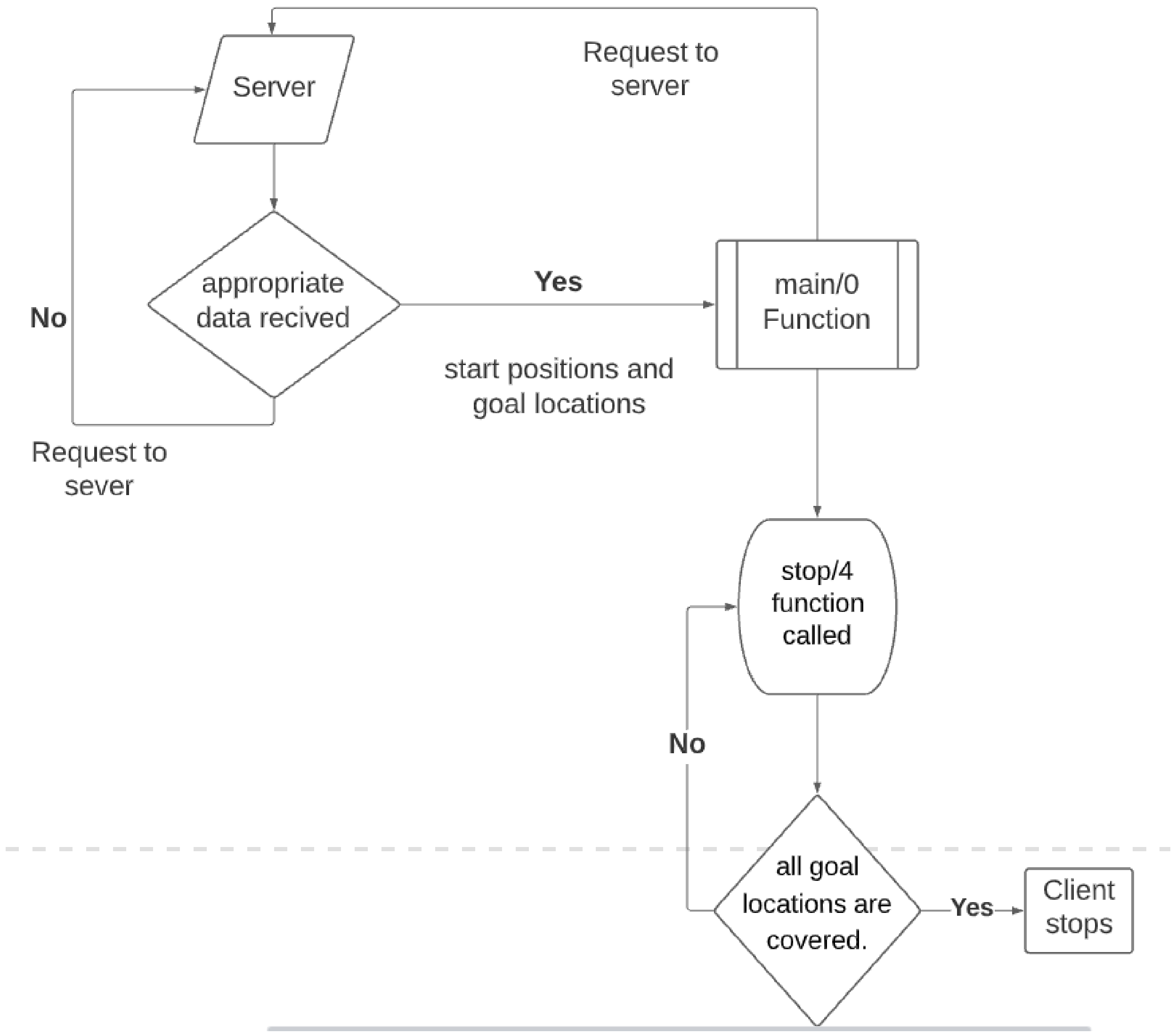
Once the goal location and the start position are received in the main function of the respective clients the stop/4 function is called passing the appropriate channel pid and the start position of the robot.
stop/4 function is called recursively to cover all the locations assigned to the respective clients by the server, all the actions taken by the robot are sent to the server via the send_robot_status/3 function defined in the phx_socket_client.ex file.

send_robot_status/3 does the following:

- Sends the robot’s position after performing various actions like turning, moving etc on the topic robot:status.
- Updates the state on the server that stores the current position of the robot which can be accessed by the client when needed via synchronous request message.

When the messages sent by the client are received in the server over the robot:status topic, the pixel location is calculated and the calculated data is sent to the ArenaLive over a topic robot:update which triggers the handle_info/2 callback in the ArenaLive module.

When obstacles are encountered by the robot, indication is sent to ArenaLive where the pixel location of the obstacles are calculated and the parameters corresponding to the obstacles are updated.



Seeding :

- Goal locations for robots are provided in Plant_Positions.csv.
- For seeding, the seeding locations are provided to the robot A
- The robot A covers all the given goal locations along with the communication with robot B over the internet.

Weeding :

- Goal locations for robots are provided in Plant_Positions.csv.
- For weeding, the weeding locations are provided to the robot B
- The robot B covers all the given goal locations along with the communication with robot A over the internet.

Communication Over Internet :

- We used a service that is **ngrok** that allows us to test our local websites by hosting them on the Internet.

- For testing purposes, we can forward our local host for a short period of time. It must be accessed via the public internet.

- A Phoenix server, by default, is hosted locally on port **4000** but it can be made available to other members of our team by hosting it via **ngrok**.

- We have first implemented Task 5 locally and then proceed for remote testing.

- To get started, we created an account for this purpose.