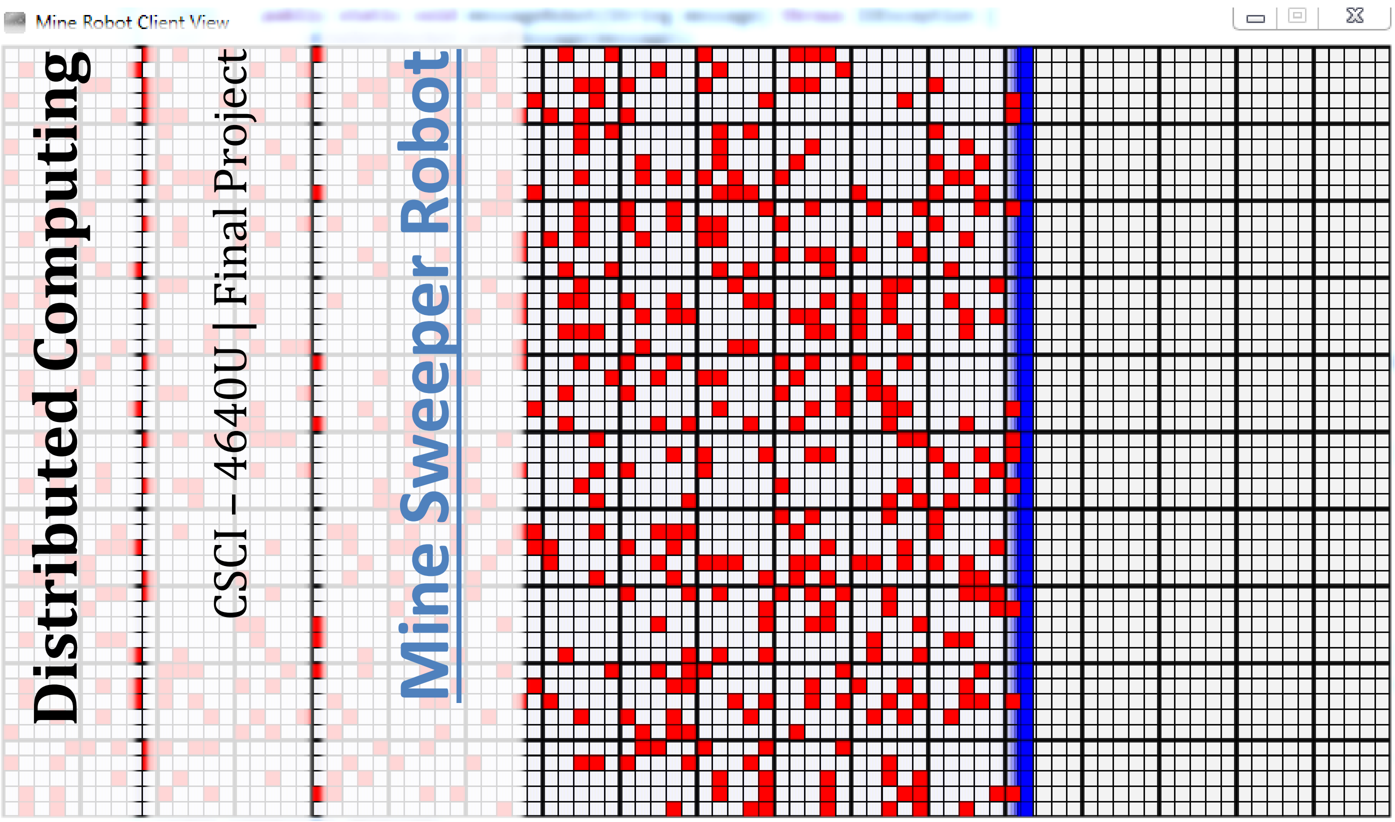
Distributed Computing

CSCI – 4640U | Final Project

Mine Sweeper Robot



A mine sweeper robot that is able to move across a field scanning a mine field with an array of sensors. Mine locations are detected and transmitted to a LIVE client view and or stored within a Database for later recall via any web enabled device.

Group 7
Allen Kaganovsky | 100389429
Matt McFadyen | 100393593

Table of Contents

Intro

This term finalizing project was to demonstrate our knowledge of Distributed Computing and how several separate systems can communicate between themselves as a single large system. Our original concept consisted of only the Live Simulation of the robot scanning a mine field; however this seemed fairly bland and unexciting. A storage and recall feature would make the application much more attractive, and could easily be done using a MySQL Database for storage, and PHP GD libraries to redraw the map within a web browser. Adding the database and history view via a web browser added yet another level to the distributed system as it required our Mine Robot Server to communicate with the Database as well as multiple web clients able to recall mine scans through HTTP Requests and the web server.

Requirements

Priority	Description
Primary	The Live Mine Scan should be visible via the client view.
Primary	The Mine Robot Server handles communication between the Robot and Client
Primary	The Mine Robot Server buffers any mine data passed from the Robot to the Client
Primary	If the Client Disconnects, the Robot will continue its scan until destination reached.
Secondary	Once the Simulation is complete, buffered Mine data on the server is stored on a Database.
Secondary	Web enabled clients can access a history of these scans via a Web Interface that generates Images based on data pulled from the Database.

Use Cases

There are four typical use case scenarios that a user may come by when using this distributed system. For the following use cases, it is assumed the Mine Robot Server is operational at all times.

Use Case 1 : User completes full scan without interruptions

1. The user positions and activates the Mine Sweeper Robot.
  1. a) If the user has not turned on the Live Client View, go to Step 2.
  1. b) If the user has already turned on the Live Client View, go to Step 3.
2. Turn on the Live Client View
3. The server will automatically begin the simulation when both Live Client and Robot are detected.
4. The simulation completes without interruption, the Robot is at the destination and the Live Client represents a map of the Live Mine Scan.
5. The server automatically stores the Mine Data once the simulation completes and is available via Web Interface.
6. The client can access the Mine History Scans via a Web Interface that retrieves a requested map scan and displays the map representation to the user.

Use Case 2 : The Client Live View disconnects prematurely

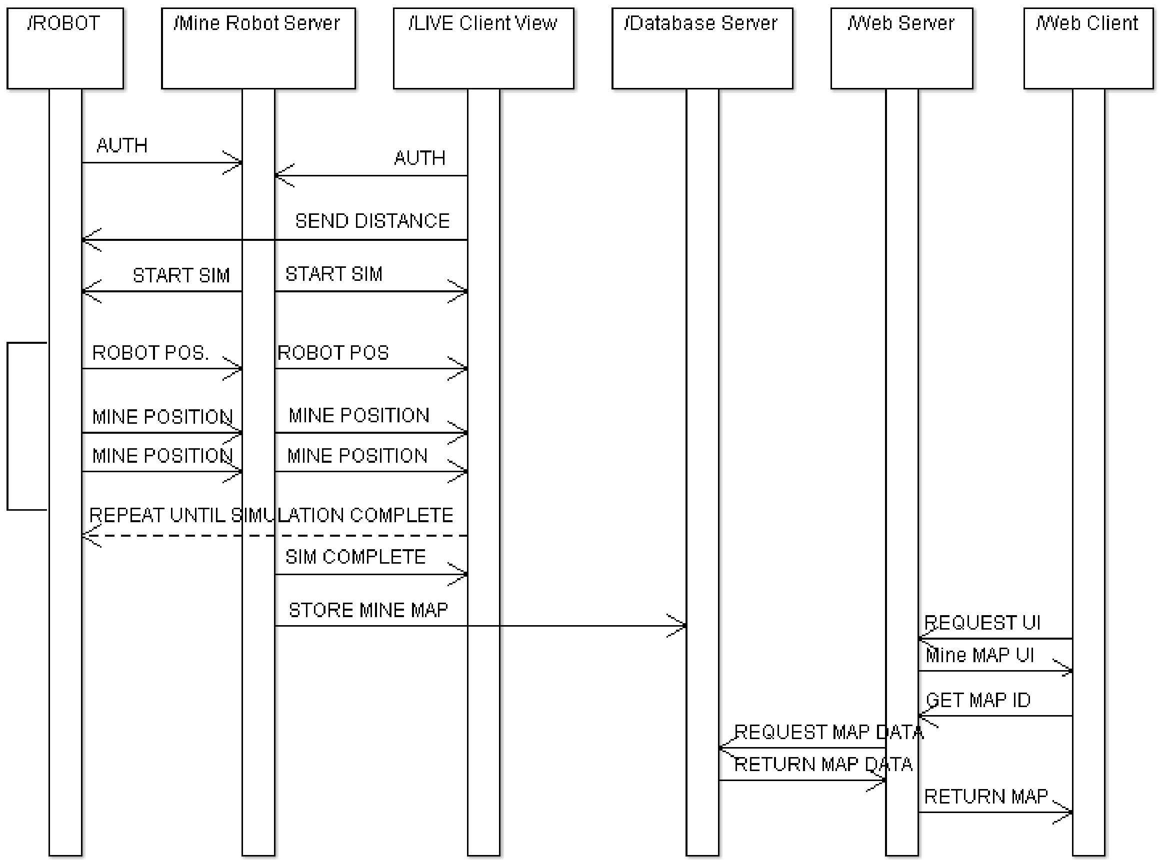
1. The user positions and activates the Mine Sweeper Robot.
  1. a) If the user has not turned on the Live Client View, go to Step 2.
  1. b) If the user has already turned on the Live Client View, go to Step 3.
2. Turn on the Live Client View
3. The server will automatically begin the simulation when both Live Client and Robot are detected.
4. During the Mine Scan, the Client Live View connection is interrupted.
5. The simulation proceeds until the Robot reaches its destination. All Mine data is still collected and stored within a Database. This Mine scan can be viewed via the Web History view.
6. The robot waits at the expected destination.

Use Case 3 : The Robot disconnects prematurely

1. The user positions and activates the Mine Sweeper Robot.
  1. a) If the user has not turned on the Live Client View, go to Step 2.
  1. b) If the user has already turned on the Live Client View, go to Step 3.
2. Turn on the Live Client View
3. The server will automatically begin the simulation when both Live Client and Robot are detected.
4. During the mine scan, the Robot connection is interrupted.
5. The Client View can see this because the robot position stops updating
6. Any buffered Mine data that had been collected before the connection was interrupted will be stored to the database as a partial scan.
7. Partial scans can be viewed through the Web interface as any complete scan.

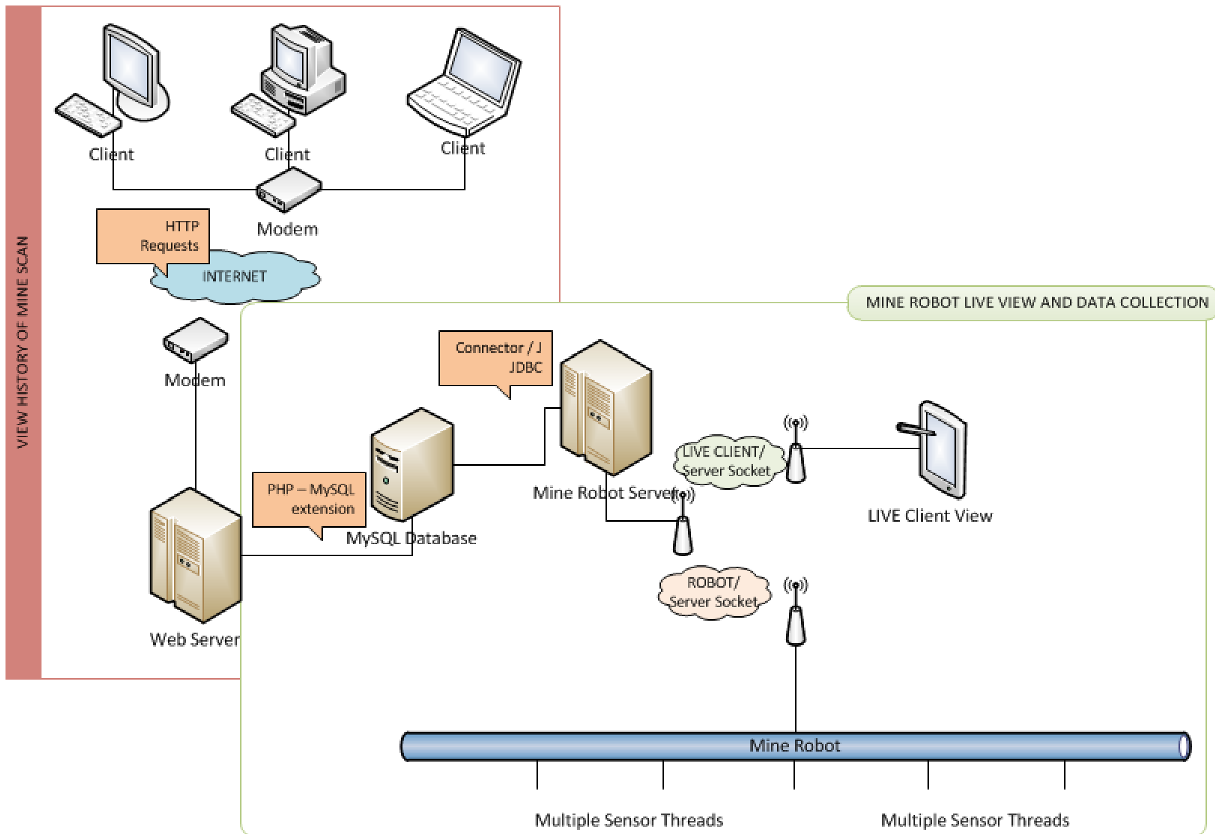
Use Case 4 : Client views Map History scan via Web Browser
1. Client uses any web browser and navigates to the web server providing the history of mine scans. 2. A scan is selected from the dropdown list of all currently available scans found in the database. 3. The Show Map History button is pressed and that selected map data is loaded within the web browser. 4. The client can choose to close the browser, print the map, or select another map for viewing.

Sequence Diagram



The above sequence diagram represents all message passing and communication between the various components found within our Distributed System. We can see that during an initial Mine scan, communication occurs between the Robot, Live Client and Mine Robot Server.

Architecture



In the above diagram, we can see all of the components involved within the entire distributed system. All of these components must talk to each other in one way or another, and may also use some sort of middleware in order to provide translation between various protocols. The communication protocols and middleware will be described in detail below.

Middleware

For our Robot Mine Sweeper distributed system, two middleware libraries have been used, and both were used in order to communicate with the MySQL server.

Connector/J JDBC or Java Database Connector was used to allow our Mine Robot Server, written in Java, to communicate quite easily with a remote MySQL Server. The MySQL group provides several connectors that allow various languages to connect to a MySQL Database including .Net, ODBC, Python, C++, C and the connector used for this project, Connector/J.

PHP + libmysqlclient PHP is the web script of choice for our web script since using the PHP GD libraries, we can easily generate an image representing the Map Scan based on stored data from the database. In order for PHP to access a MySQL Database, it must be installed or recompiled with MySQL Client libraries in order to form a connection between the web server and the MySQL server.

Communication Protocols

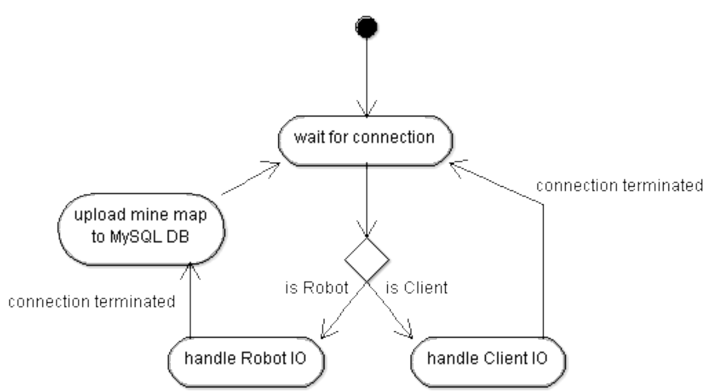
Socket Protocol was used to handle the communication between the Robot, Mine Robot Server and Live Client view. The socket protocol allowed for us to create communication channels to stream data via message passes. We choose to use Sockets primarily as it was a very easy to detect a client disconnect during a data stream connection. This is an important feature with the Mine Robot Server as once it detect the Robot had disconnected, it assumes the simulation is complete and uploads the Mine Data to the MySQL Database. The Server doesn't particularly know if the Robot reached its destination or not, it simply assumes that if the Robot disconnects, it's because it has reached its destination and turned off, or destroyed by a mine.

HTTP Requests is another major protocol used for our distributed system as it is the backbone that allowed clients to view history of map scans via a web browser. A client initially performs an HTTP GET Request to get the web interface they will be working with. The client then issues a HTTP POST command to instruct the server of which map to generate. After which, the page is issued back to the client with the generated image.

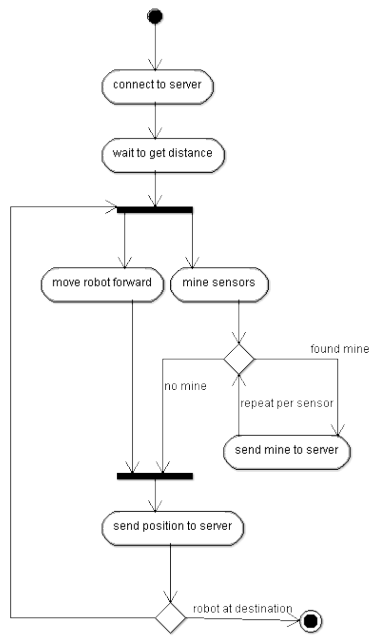
Behavioral Design

State and Activity Diagrams

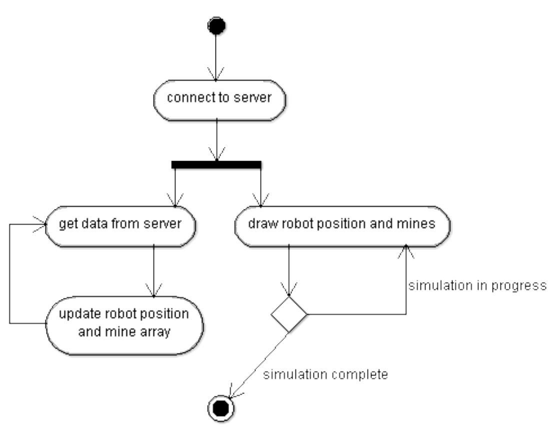
Mine Robot Server : Initially waits for a client or robot connection, only one of each kind is allowed at a time. If a robot or client disconnect from the server, the server is able to wait for another client, while still dealing with the current simulation if the Robot is still connected and transmitting Mine data.



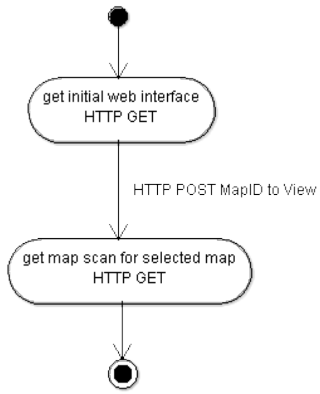
Mine Robot : Continually feeds the server with its position and found mines, the server then relays the appropriate information to the Live Client View. The initial distance the robot receives before the simulation begins is given by the field length specified within the client view.



Live Client View : Continually reads data from the server and draws the updated robot position and any mines found by the robot.



Web Application to view Map Scan History : Any web enabled client can access the server via a web browser. This initial request is a GET request where the server sends the client the GUI of the application. The client then GETS an updated map scan view by POSTing to the server which map scan to generate.

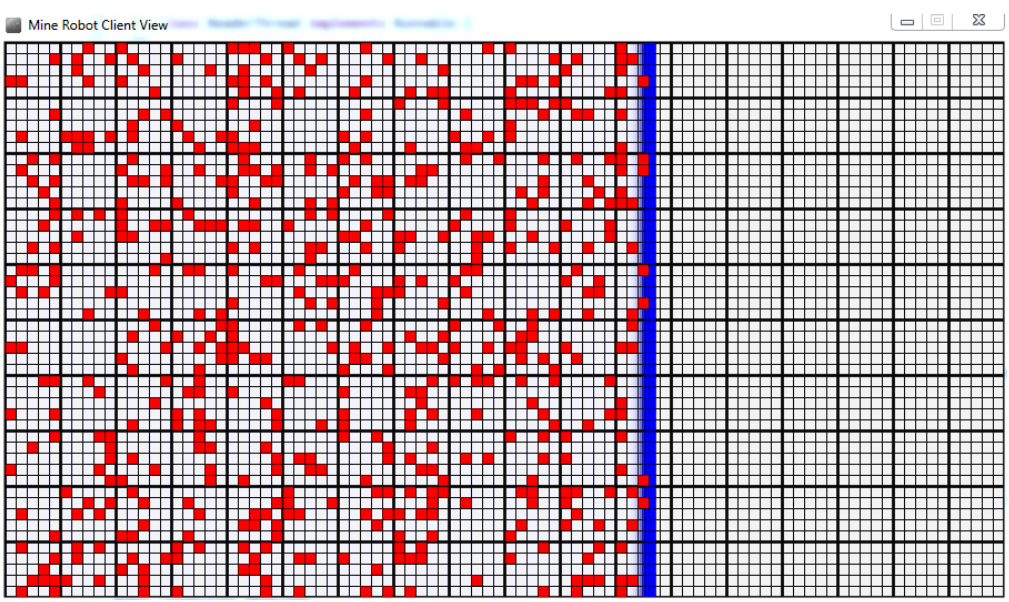


Proof of Concept

Demonstration

With each sensor set to detect a mine by having a 10% chance of detecting a mine per grid box, we can run the following simulation.

Live view while scan is in progress… The blue vertical bar represents the Robot, where a sensor exists for each grid box on the robot. As the robot moves across the field, sensors fire off messages to the Server when a mine is detected.

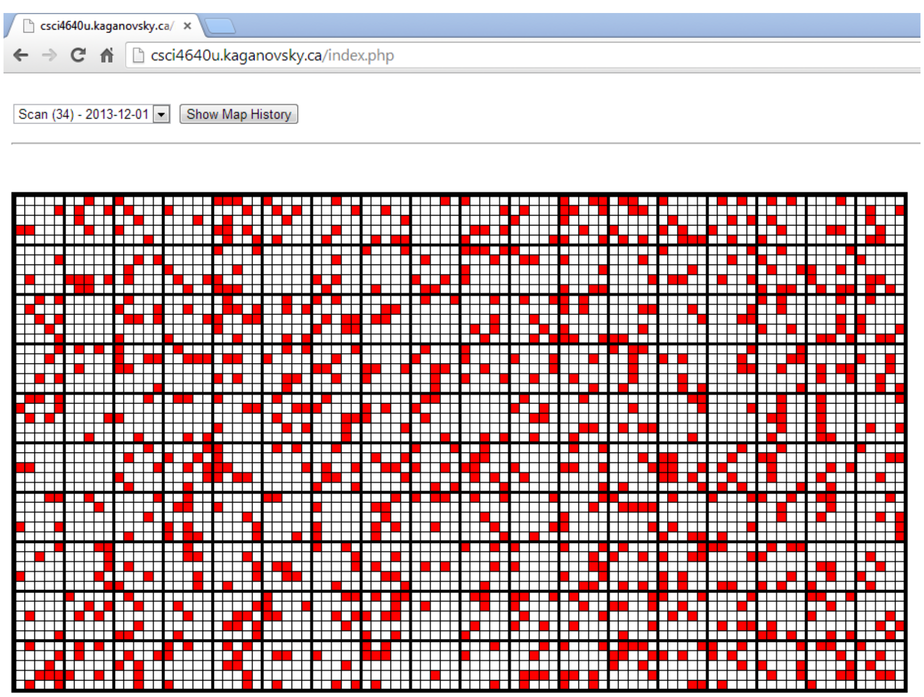


Once the scan completes, the blue bar is hidden off the screen and a full Mine map is presented to us in the Live View. This scan is encoded by the server by generating a string of all the mine locations and storing them on a MySQL Database server.

	id	minestring	date
Edit Copy Delete	1	-10,40.-10,100.-10,380.-10,450.0,470.0,60.0,250.0,...	2013-11-26
Edit Copy Delete	2	0,390.0,320.0,300.10,420.10,130.10,460.20,40.20,16...	2013-11-26
Edit Copy Delete	3	0,150.0,450.0,310.10,10.10,160.10,300.10,430.20,40...	2013-11-26
Edit Copy Delete	4	0,70.0,310.0,50.0,140.0,380.0,420.10,30.10,110.10,...	2013-11-26
Edit Copy Delete	5	0,130.0,40.0,340.0,460.0,310.0,160.0,410.0,450.10,...	2013-11-27
Edit Copy Delete	6	0,250.0,320.0,480.10,110.10,460.10,210.10,50.10,90...	2013-11-27
Edit Copy Delete	7	0,30.0,40.0,60.0,220.0,440.10,380.10,230.10,400.20...	2013-11-27
Edit Copy Delete	8	0,380.10,340.10,230.10,150.10,460.10,390.10,160.20...	2013-11-27
Edit Copy Delete	9	0,160.0,180.0,190.0,230.0,380.0,430.10,470.10,280....	2013-11-27
Edit Copy Delete	10	0,300.0,290.10,310.10,340.10,80.10,60.10,250.10,16...	2013-11-27
Edit Copy Delete	11	0,90.0,80.0,40.0,0.0,20.0,30.0,60.0,70.0,100.0,310...	2013-11-27
Edit Copy Delete	12	0,140.0,230.0,60.0,100.0,30.0,200.0,10.0,80.0,450....	2013-11-27
Edit Copy Delete	13	0,140.0,230.0,60.0,100.0,30.0,200.0,10.0,80.0,450....	2013-11-27
Edit Copy Delete	14	0,30.0,70.0,110.0,150.0,190.0,230.0,270.0,310.0,35...	2013-11-27
Edit Copy Delete	15	0,20.0,140.0,410.0,70.0,50.0,460.0,310.0,240.0,270...	2013-11-27
Edit Copy Delete	16	0,140.0,200.0,440.0,20.0,100.0,290.0,450.10,0.10,1...	2013-11-27
Edit Copy Delete	17	0,140.0,200.0,440.0,20.0,100.0,290.0,450.10,0.10,1...	2013-11-27
Edit Copy Delete	18	0,140.0,20.0,220.0,210.0,130.0,10.0,100.0,260.0,50...	2013-11-27
Edit Copy Delete	19	0,140.0,20.0,220.0,210.0,130.0,10.0,100.0,260.0,50...	2013-11-27
Edit Copy Delete	20	0,260.0,360.0,310.0,370.0,440.10,140.10,230.10,200...	2013-11-27
Edit Copy Delete	21	0,0.0,120.0,20.0,380.0,230.0,480.10,360.10,350.10,...	2013-11-28

This database is the backbone for the web application used to view mine scan history maps.

Through a web interface, the client is presented with a dropdown box with all of the available scans to view. Once a client selects a scan and clicks View Map Scan, the mine string is pulled from the database at the appropriate row, decoded into x,y pairs and then drawn on a similar grid view as the Live client view.

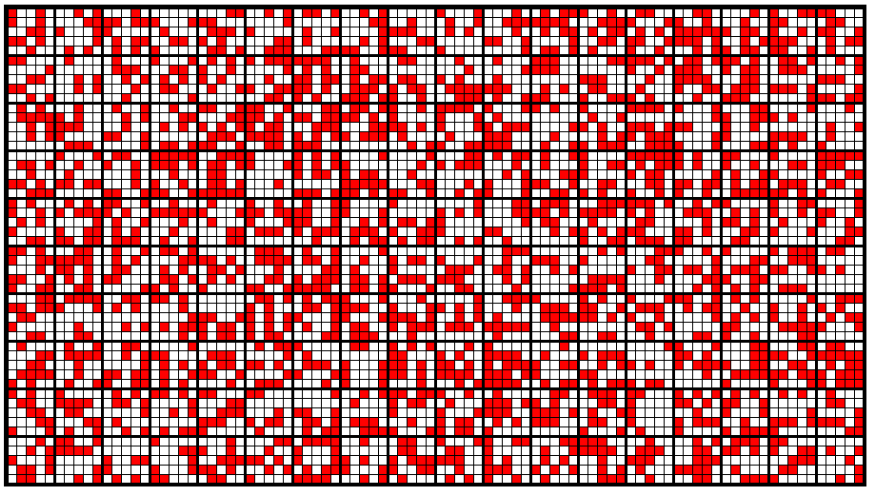


Tests

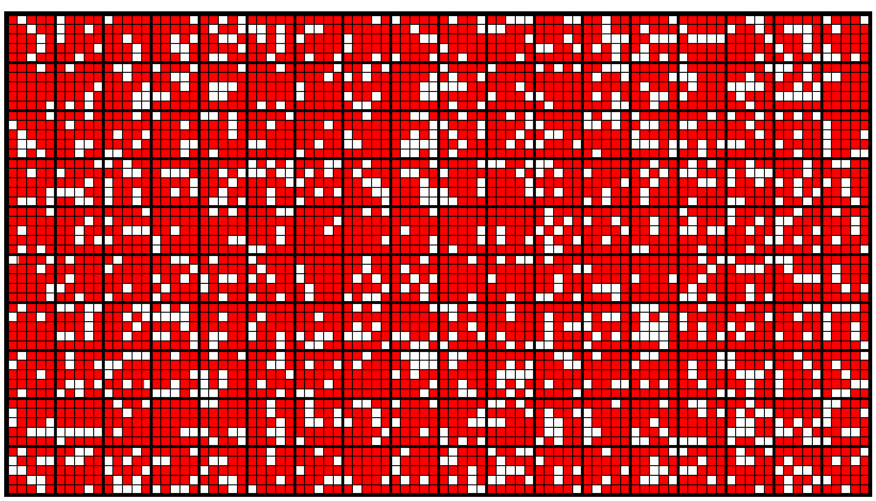
We have done several tests by varying the degree by which a sensor will detect a mine per grid box.

We have tested several scenarios including 10%, as seen in the demonstration, 25%, 50%, 75% and 100% chance that a mine will be detected.

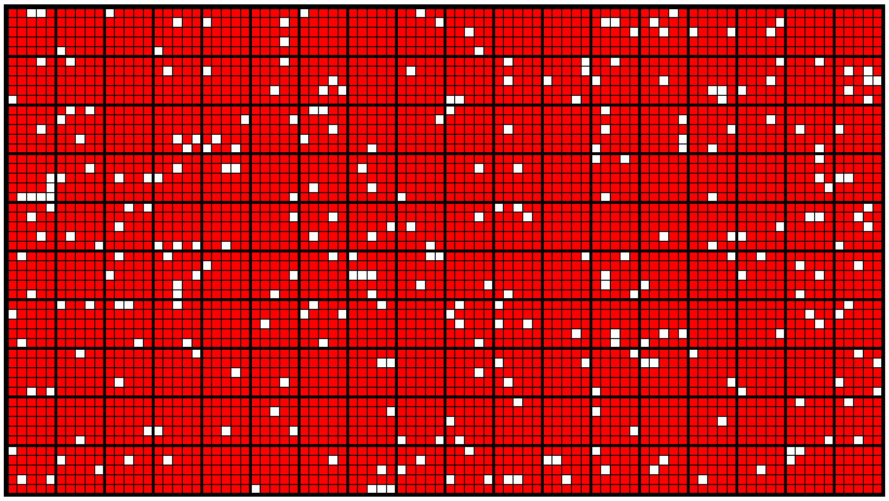
25%:



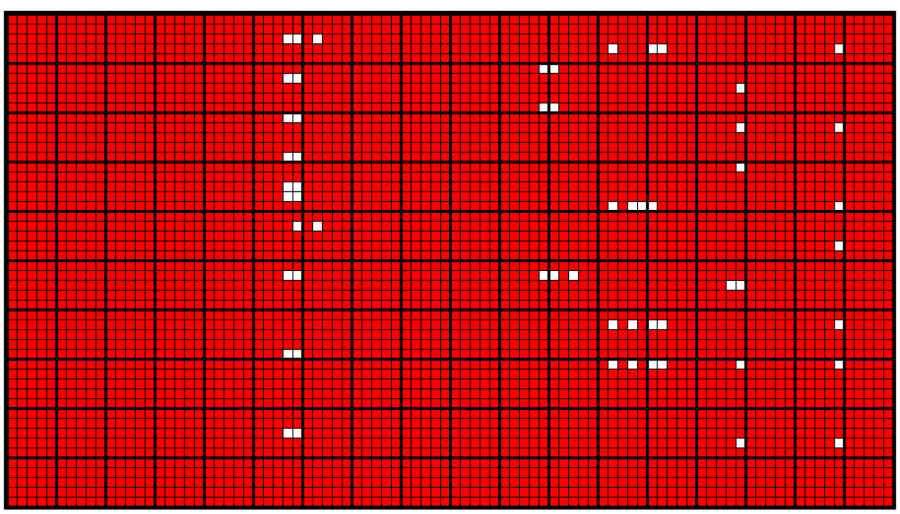
50%:



75%:

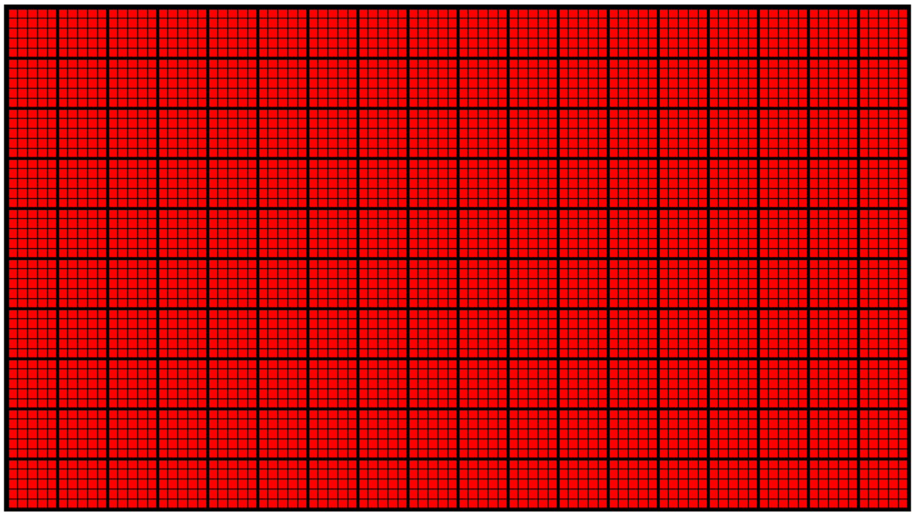


100%: Not Synchronized (We can see some gaps were messages were blocked or lost)



This issue was fixed by synchronizing the message send command from the robot to the server as well as decreasing the sleep time of the sensor loop.

100% Synchronized



We can see that our system is able to handle the load now that our message commands are synchronized in order to have 50 simultaneous threads try and access the method at one time.

Furthermore, the implementation of the robot itself could be changed in that the robot does not proceed from its current location until all of the messages have been sent for that location. Once the robot completes passing all of the mine locations for that particular area, the robot then moves forward.

Conclusion

We have demonstrated that we were able to build a distributed system using several components, protocols and middleware to have those components communicate amongst once another. Our system utilizes basic Java Sockets for our remote Robot and Live Client to communicate between each other with an intermediate server in the middle. This intermediate server stores collected mine data and uploads the array to a MySQL Database Server. In order for the Java server to communicate with the MySQL Server, a middleware library called Connector/J was used. Furthermore, the client can then use a web browser and access a history of mine scans. Any web browser performs HTTP Get requests in order to obtain a web page. The client is then able to get a generated mine map by posting which map to view, and getting the accompanying result. This is done through HTTP POST and GET requests. The web language used to generate the mine map is PHP, and PHP too requires MySQL libraries installed in order for the language to be able to communicate with the MySQL Database Server.

All in all, our system is able to conduct a live mine scan while viewing the current scan in real time, and also have the ability to view a history of these scans view a web browser.