Multi-Robot Navigation System with Collision Avoidance

Asher A. Hensley

1. Introduction

This report describes the design of a server controlled multi-robot navigation system in SystemC. The main objective of the system is to simultaneously guide multiple robots through a 2D space containing barriers and obstacles. The server has been equipped with special collision avoidance logic to which is used to control the speeds of the robots and to avoid deadlock conditions. We first give a high level overview of the system including block diagrams, modules, and processes. We then describe and give results for our design verification test, followed by some concluding remarks.

2. System Overview

2.1 Block Diagram

A high-level system diagram for a single robot configuration is given below in **Figure 1** for the sake of brevity. The diagram can be extended to realize multiple robots by imagining additional robot/sensor blocks connected to the server block.

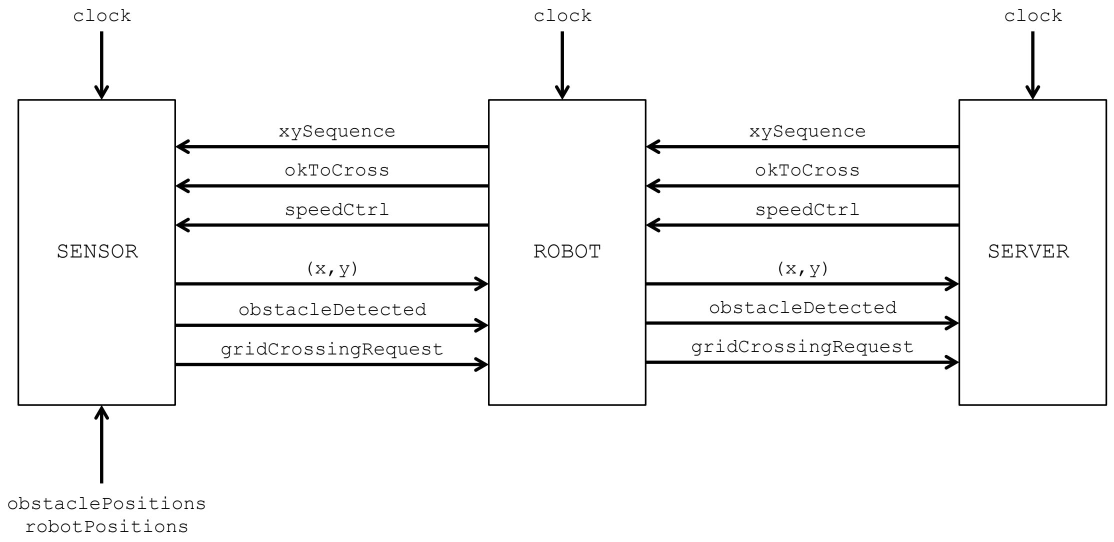

Figure 1: System Diagram

2.2 Modules & Processes

- `SC_MODULE(server)`
  - `SC_THREAD(sendPathMessage)`
  - `SC_METHOD(checkConflictRobot1)`
  - `SC_METHOD(checkConflictRobot2)`
  - `SC_METHOD(checkConflictRobot3)`
  - `SC_METHOD(updatePointers)`
  - `SC_METHOD(speedCtrl)`

- `SC_MODULE(sensor)`
    - `SC_METHOD(updatePathFifo)`
    - `SC_METHOD(updateTrajectory)`
    - `SC_METHOD(checkPosition)`
    - `SC_METHOD(checkObstacles)`

- `SC_MODULE(robot)`
    - `SC_METHOD(relayGridRequest)`
    - `SC_METHOD(procServerPermissions)`
    - `SC_METHOD(procObstacleAlert)`

- `SC_MODULE(obstacle)`
    - `SC_METHOD(sensorPositionUpdate)`

3. Design Verification Test

3.1 Test

The test has been designed to verify the server speed control logic is functioning correctly by placing robots on trajectories that will result in deadlock if speed control is not used. The test includes 3 robots and 2 obstacles on the paths shown in **Figure 2**.

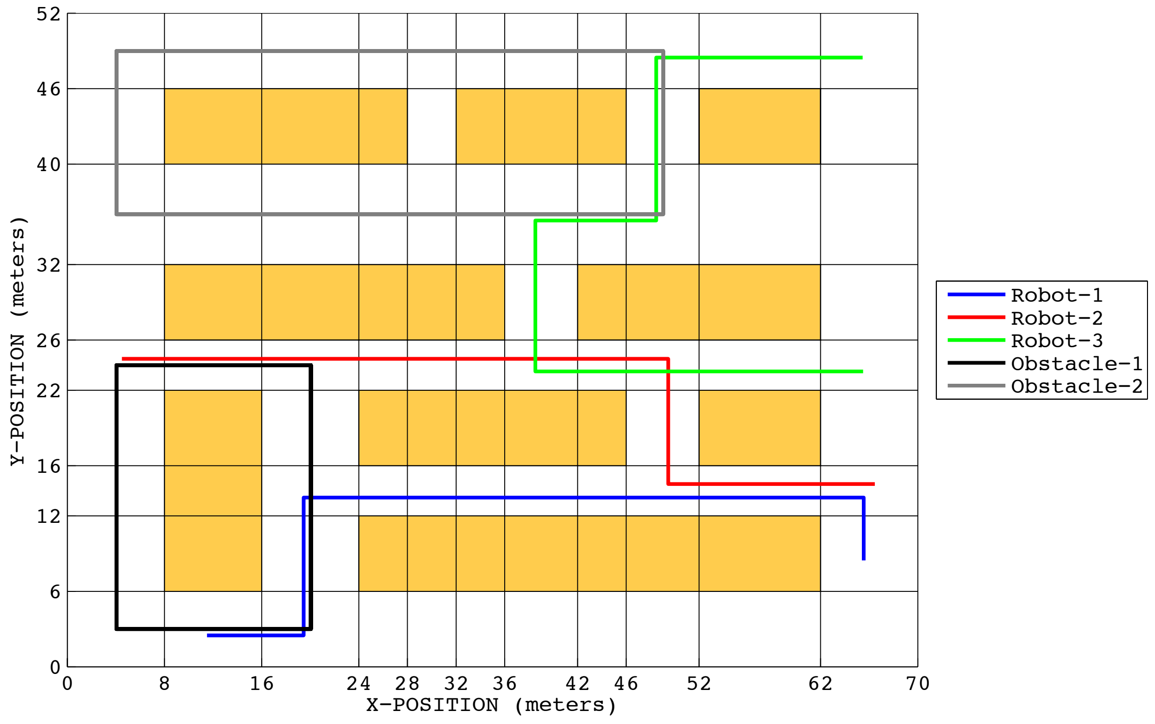

Figure 2: Design verification test scenario (note: robot positions have been slightly offset in the plot for visualization purposes).

3.2 Failure Mode

We began by running the test with the server speed control disabled. In this configuration, robots travel at a constant speed unless they encounter an obstacle, in which case they stop. The results for this run are shown in **Figure 3**. There is clearly a gridlock condition between robot-2 (red) and robot-3 (green). For this test to succeed, robot-3 must clear the intersection at (x,y) = (39,24) before robot-2 arrives. Although not shown, a similar problem arises between robot-1 (blue) and robot-2 in the intersection at (x,y) = (49,14).

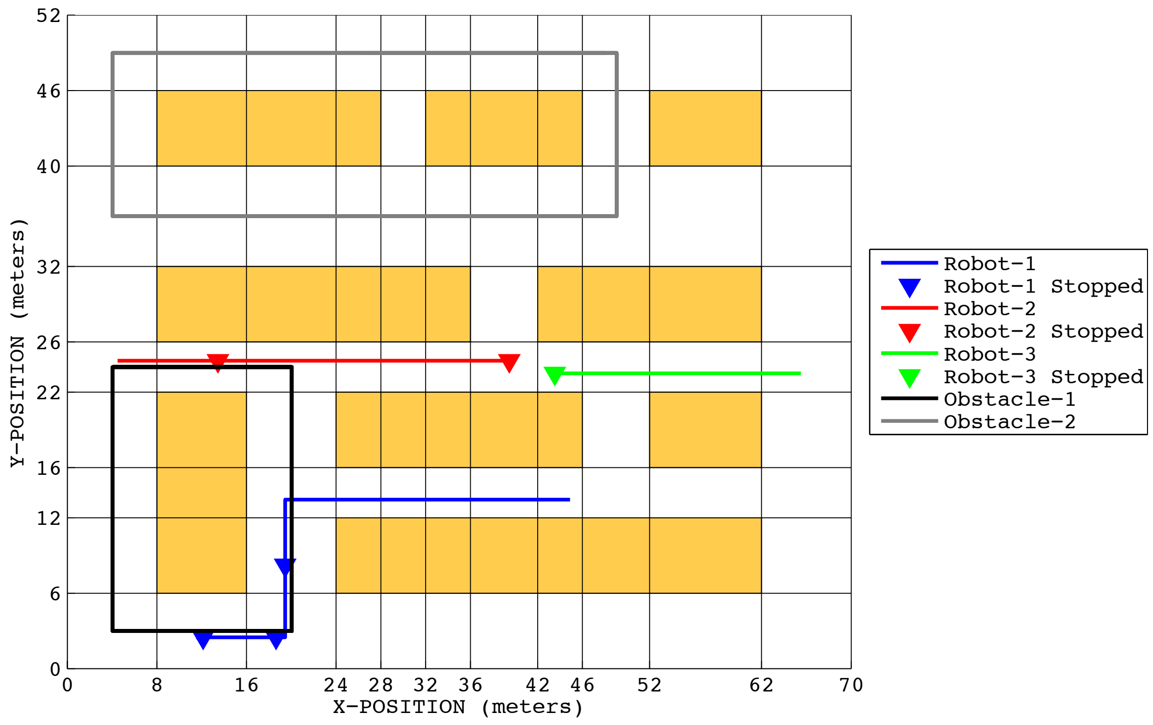

Figure 3: Design verification test scenario (SPEED CONTROL DISABLED).

3.3 Verification Results

Next we re-ran the test scenario with speed control enabled (see **Figure 4**). From this plot we can clearly see that using adaptive speed control was able to avoid the deadlock condition observed in **Figure 3**. The speed of each robot as a function of time is shown in **Figure 5**.

4. Conclusion

Based on the results of our design verification testing in phases 1, 2, and 3 we can conclude that the design is functioning correctly. The adaptive speed control was the key enhancement to overcome deadlock conditions likely to occur in real-world scenarios. Future work could include adding alternate route logic and to give the robots the ability to make u-turns.

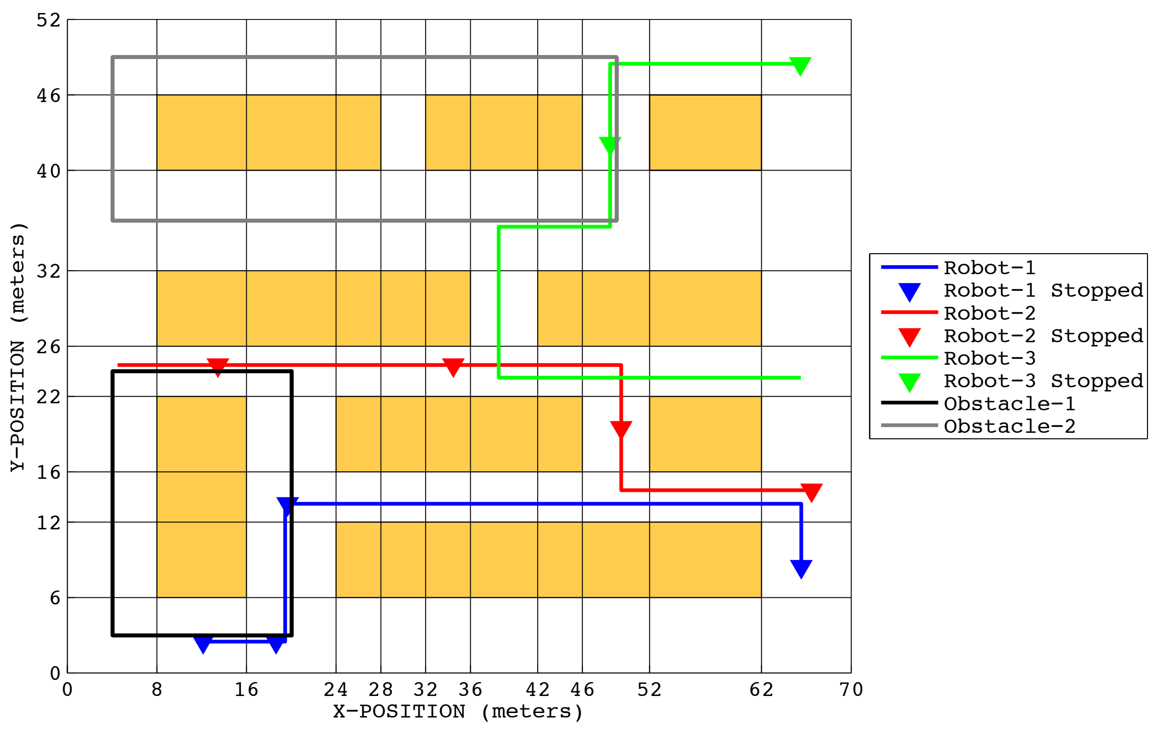

Figure 4: Design verification test scenario (SPEED CONTROL ENABLED).

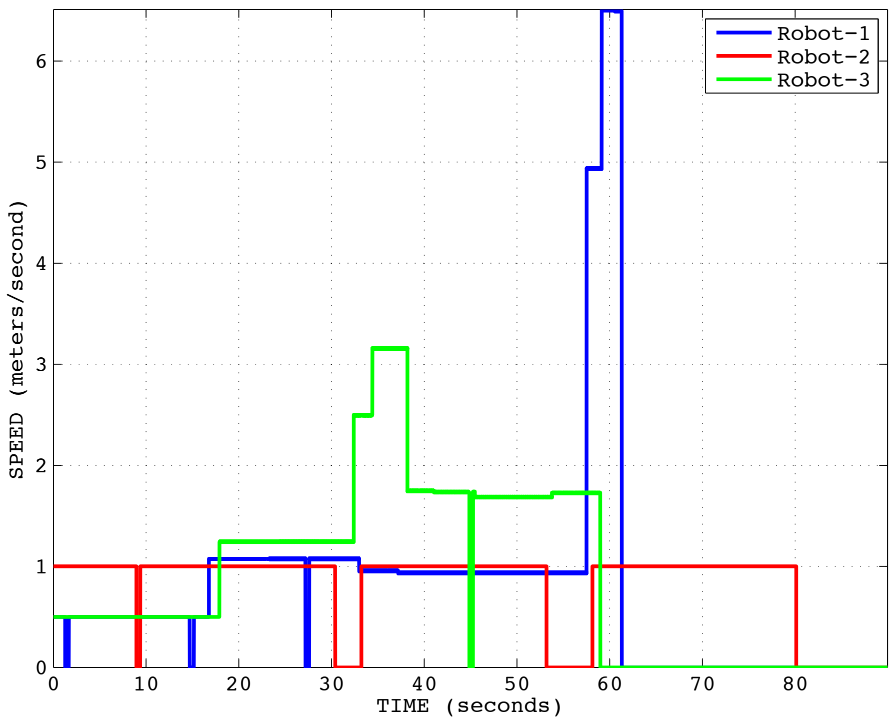

Figure 5: Robot speeds during test scenario.