Summary Report: Robot Navigation in Confined Areas Phase 3

I. Processing Module

The processing module defines and implements the path handling, robot handling, obstacle handling, and communication handling processes. The map data structure is defined within the processing module as it contains the x and y coordinates that correspond to the individual 60 grids within the map. The map data structure also contains information on the grids that are north, south, west, and east of each individual grid. The navigation behaviors of robots are simulated within the processing module as well. The processing module has its prc_update() process that constantly updates the position of each robot based on the status of the robot and according to the speed (2m/s) and path of each individual robot that's sent by the server module every clock cycle. The statuses of the robots are defined below:

```
// ************************************
// Status Value from Robot (Processing) to Server
// Status 0 -> STOPPED1 (stopped due to obstacles)
// Status 1 -> RESTART (ask server to started after obstacle clearance)
// Status 2 -> CROSSING (just before the boundary crossing)
// Status 3 -> STOPPED2 (stopped at the boundary due to no Ack from Server)
// Status 4 -> CROSSED (this will tell server to update server's table)
// ************************************
```

During each cycle, the position of the obstacle will be updated as they travel at a constant speed of 4m/s around each of the six islands. If at any point during the process that the robot's coordinates are within 3 units of the obstacle's coordinates, the status in the main table of the respective robot index will change to 0, indicating that the robot has stopped due to obstacles and the modified index will be set to 1. The tx_counter will be incremented every time a modification has been made to the main_table. If the robot gets close to the boundary of the grid (10% of the grid size), the status of the main table will be changed to 2, indicating that the robot is currently crossing the boundary and the modified index will be set to 1. Upon crossing the boundary, the status of the main table will be changed to 4, indicating that the robot has crossed the boundary. At the end of the prc_update() process, the prc_tx() process will be notified through the sc_event. The prc_tx() process then scans the updated table for any modifications made and then transfers the status to the tx_table. The data containing information of the status of each robot within the table will be forwarded to the robot module from the processing module while also outputting the flag into the prc_rx_p() process of the robot module.

II. Robot Module

The robot module is responsible for forwarding the data sent from the processing module's tx_table from the prc_tx() process into the main table and then forwarding the modified data into the server module's rx_table from it's prc_tx() process. The robot module is also responsible for transferring the data from the server module to the processing module when the status value is updated by the server module is to be sent to the processing module through the robot module.

III. Server Module

The server module provides the path as well as the speed of each robot for the processing module through FIFO as the times at which the path is sent is based on the robots' individual start times. The server module determines if the robot should cross the boundary or resume after an obstacle is cleared by checking if obstacles are within the current and next grid. The status will be stored in the tx_table and the modification index will be set to 1. The data containing information of the status of each robot within the table will be forwarded to the robot module from the server module while also outputting the flag into the prc_rx_s() process of the robot module. The statuses that could be sent from the server module is shown below:

```
// ************************************
// Status Value from Server to Robot (Processing)
// Status 5 -> OK1 (ok to cross)
// Status 6 -> OK2 (ok to move)
// Status 7 -> STOP1 (stop, do not cross, do not move)
// Status 8 -> STOP2 (do not move)
// Status 9 -> RESUME (resume from previous stop)
// Status 10 -> 10 SPEED (sending speed value via FIFO) – Phase 2
// Status 11 -> PATH (sending path sequence via FIFO) – Phase 2
// ************************************
```

In addition, the server module will also check whether the robot is about to reach an intersection (i.e a grid with multiple ways for the robot to move) and then allow the robot that is closer to that intersection to move through first while the other robot waits. This allows the robot to not reach a deadlock due to multiple robots waiting for each other during an intersection. The server is also responsible for controlling the speed of the robot. Each robot will increase their speed if there are no obstacles in the way all the way until 2m/s in 40 speed intervals of 0.05m/s.

IV. Main File

The main file simply connects the inputs and outputs of the robot module, processing module, and server module together. A speed variation plot is also generated by creating a vcd trace file that contains information about the speed of each robot throughout the time it travels through the paths sent.