Design of the Maze Explorer

Wenguang Wang

1 Preface

This is only a draft. Everything in this document is under everyone's discussion. Our project team includes following people:

Prachi	pra961@cs.usask.ca		
Steve	akira0@home.com		
Tony	mishralk@hotmail.com		955-3526(H)
Wayne	wew036@cs.usask.ca	966-8654(O)	955-8057(H)

2 System Architecture

Figure 1 is the architecture of the system.

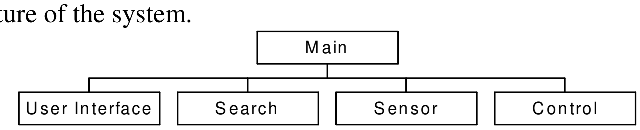

Figure 1 System architecture

- **Main Module** controls all sub-modules of the system, schedule the events like reading sensors, calling the search modules and sending control command to the robot.
- **User Interface** provides a friendly graphic user interface, to start, stop the robot, and view the map and the robot activities dynamically.
- **Search Module** makes decisions on where and when to move based on certain search strategy and the input of sensors.
- **Sensor Module** read the input of rotation sensors, touch sensors, and light sensors, convert them into formats that can be easily understand by the other modules.
- **Control Module** receives commands from other modules and sends them to the robot. These commands may include moving the robot at a certain speed, stopping, turning a certain angle.

3 User Interface

The user interface can control the robot, view the current map. Figure 2 shows a sample.

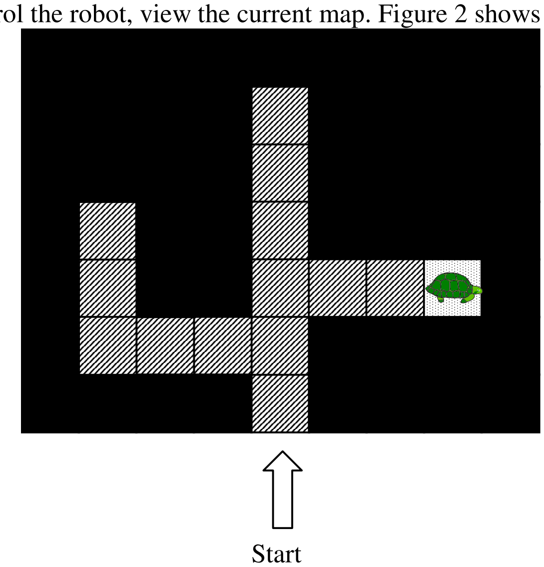

Figure 2 User Interface

4 Detailed Tasks

For doing our project successfully, we need following tasks:

1. *Project goal.* This is what we want to do in this project. All of our work will follow this goal. Any effort that is not helpful to our goal should have low priority. We need to define a basic goal, and some additional features. The basic goal includes the most basic functions that must be implemented in this project. This guarantees us to get a reasonable (maybe not high) mark. If we have extra time, we can implement the additional features and hope to get a higher mark.
   1.1. *Basic goal.* The basic goal is to construct a robot that can explore a maze, and find a route to the golden point. The maze consists of many squares with the same size, but different colors. Every square should be big enough to contain the whole robot. A black square is an obstacle that the robot cannot go through. A white square is the road of the maze. A gray square is also the road, and it indicates that there are branch paths beside this square. The golden square is the goal of the maze.
   1.2. *Additional features.* Various additional features should be sorted by their priority.
   - Use touch sensors to detect the obstacle in addition to sense the black color as the border of the path.
   - Let the light sensors "look" at the side, and use obstacles as the border of the path. Therefore, we don't need black and gray color on the map. The robot can find the way by the light sensors and touch sensors.
   - Let the robot go back to the start point in the shortest path after it reaches the golden square.
   - Change the robot to a truck, and put a cube in the golden square. When the robot find the golden square, it loads the cube and carries it back to the start point in the shortest path.
   - Change the path when the robot is exploring the maze. We need to change the path fast when the robot is working.
2. *Interface definition.* Interface definition defines the interface between modules. That is, what functions will a module provides, in what parameters. After this definition is defined, everyone can assume the other module provides such interfaces, and begin his or her own programming. At the same time, he or she must follow the interface definition, provides the exact interface to other modules. This is extremely important for our cooperate work.
3. *Programming and test of the Main Module.*
4. *Programming and test of the User Interface Module.*
5. *Programming and test of the Search Module.*
6. *Programming and test of the Sensor Module.*
7. *Programming and test of the Control Module.*
8. *Design and construct of the robot.* This is not a trivial work. It is not easy to construct a pretty, solid robot.
9. *Design and construct various maps for testing and demonstration.* Maybe build some squares with different colors and put them together.
10. *Design and programming of test modules for unit testing.* When every member is working on there own module, he or she should design and program a test module to test his or her own module before any integration with other modules. This is very important for the system's stability and robustness. Detecting most bugs of your module before the integration can save us much time.
11. *Integrate and test the whole system.* This should NOT be the last step. Some design defections are often found during the integration stage. To do integration earlier can help find these defections earlier and correct them in a lower price. Another benefit for doing integration earlier is that we can always keep a system that can run, although its function is not complete. Therefore, we can avoid making an unrealistic goal but cannot finish it at last. When everyone finishes a preliminary module with some basic functions, we need to put them together and test them. Our later work should be based on this integrated system.
12. *Project documentation.* An important part to give us good marks.
   12.1. *Internal documentation.* Every one who develops a module is responsible to write the internal documentation for that module. The internal documentation should include the interface definition, algorithm description, usage and other documents that Eric may require.
   12.2. *External documentation.* This includes the user's manual and other documents that Eric may require.

5 Work plan

1. Define the basic goal. Define additional features and order them. Every following work should aim at the basic goal and the additional features.
2. Assign the detailed tasks to everyone.
3. Define the interface definition.
4. Program a basic framework, which includes the basic main module and a very basic user interface. After this has been done, we can program other modules under the framework at the same time.
5. Everyone works parallelly on his or her own module. The module must provide the interface that is defined in the interface definition. The developer must test his or her own module carefully and thoroughly before the integration.
6. After every module has its basic functions, integrate them together. Correct any problems under the integration.
7. Everyone continue to finish his or her module under the integrated system. We periodically merge everyone's new code and test it. Versions must be controlled carefully.
8. After the basic goal is reached, make a thorough test. Begin to write documentation. At the same time, implement additional features in order.
9. When the due day of the project approaching, stop adding new features. Continue to test the system, and furnish the documentation during this time.
10. Make a demonstration plan that can show the power of our robot mostly. Select a person to practice the demonstration several times.

6 Task arrangement

This is a possible arrangement:

- Person 1: User interface module, user's manual.
- Person 2: Search module, main module, interface definition.
- Person 3: Sensor module.
- Person 4: Control module.
- Person 3&4: Design and construct the robot; construct maps.
- Everyone will be involved in task 1, 2, 10, 11, and 12.

I like the search module. I can write the interface definition for our discussion. Person 3 and 4 need have their own computer, because they need to test their program by the real robot.