Lisa Anthony
Robot Building Lab
Grad Project: Software/Experiment Design Document
Due November 28, 2001

Graduate Students: Please create the design document for your final project. Mention all the software you are planning to use (reuse); mention libraries received with interactive-C, programs you are using written by your (or some other) team, and programs supplied by someone else. Mention all the modules you are going to write, their functions and responsibilities. Include pseudocode, diagrams, flowcharts or any other visual documentation.

1 Task

I propose to investigate the accuracy of the sonar sensors we have been using in class; many groups received readings which did not make sense, given certain assumptions that were made about the operation of these sensors. To attain a more reliable sonar sensor would be both interesting and practical for future use of these sensors.

2 Decision Tree: Experimental Conditions to Study

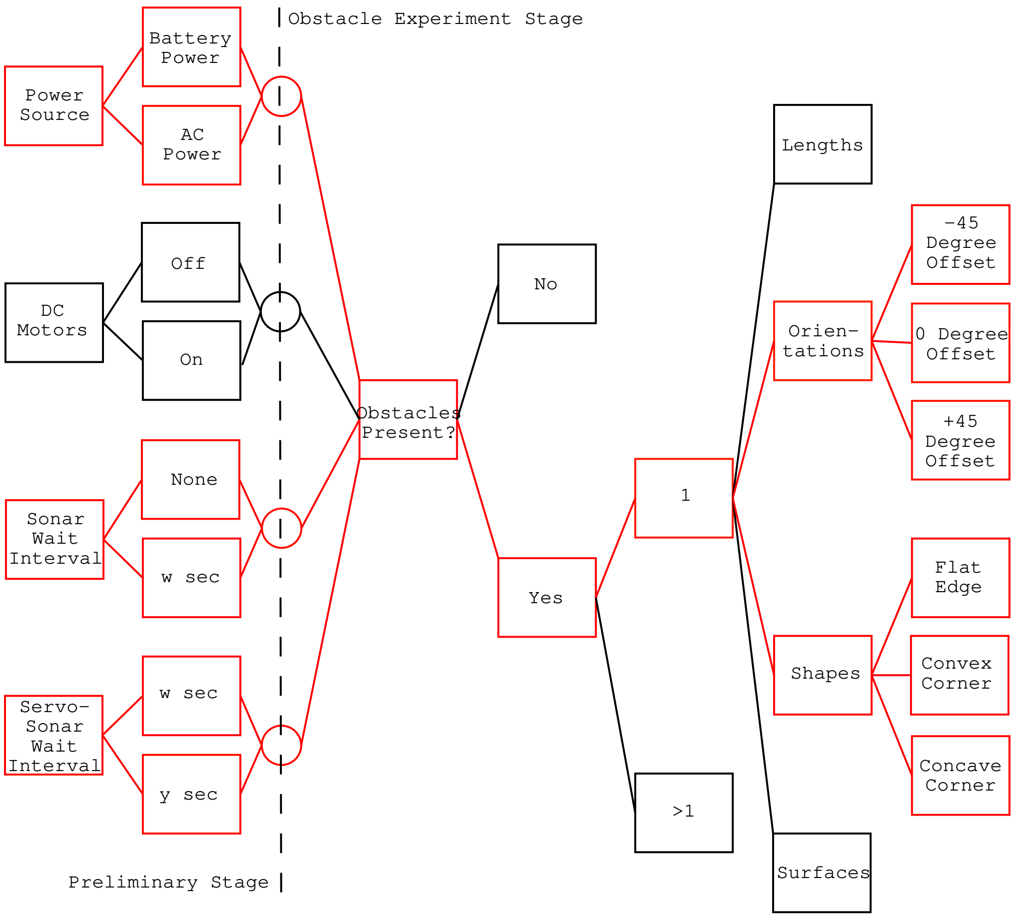

Figure 1: The decision tree illustrating what variables may affect the sonar readings. Paths to be explored during this project's experimentation are in red.

3 Software Needed

3.1 Interactive-C Libraries

- *sonar.c*—for sonar control
  - *sonar_init()*
  - *sonar_closeup()*
  - *sonar_sample()*
- *servo.c*—for servo motor control
  - *servo_on()*
  - *servo_deg()*
- *serialio.c*—for serial communication to a PC
  - *disable_pcode()*
  - *serial_putchar()*
- *printdec.c*—for printing decimal numbers via serial communication to a PC
  - *printdec()*

3.2 Control Program Modules

3.2.0.1 Serial Communication Module In previous labs, my team has built methods to use in taking collected sensor data and dumping it to a serial line, which I will also use in this project. Our team has also altered the printdec.c library to allow it to print negative numbers and zero values, which seem to be a problem in the current library.

3.2.0.2 Experimental Control Modules Figure 1 showed the paths to be explored during the experimentation phase of this project. Any decision nodes in red indicate an experiment which must be run, possibly in combination with others (the other nodes along the same path). All of these paths must be followed for both sonar sampling methods (*sonar_closeup()* and *sonar_sample()*. What follows are design diagrams for the control module needed for each version of the experiments.

Experiment 1: Power Source *Variables: Battery Power vs. AC Power*

Part a: Battery Power

1. Set up obstacle within maximum range.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Part b: AC Power

1. Plug robot into AC current.
2. Set up obstacle within maximum range.
3. Sweep servo from 0 to 180 every 2 degrees.
4. Take a sonar reading.

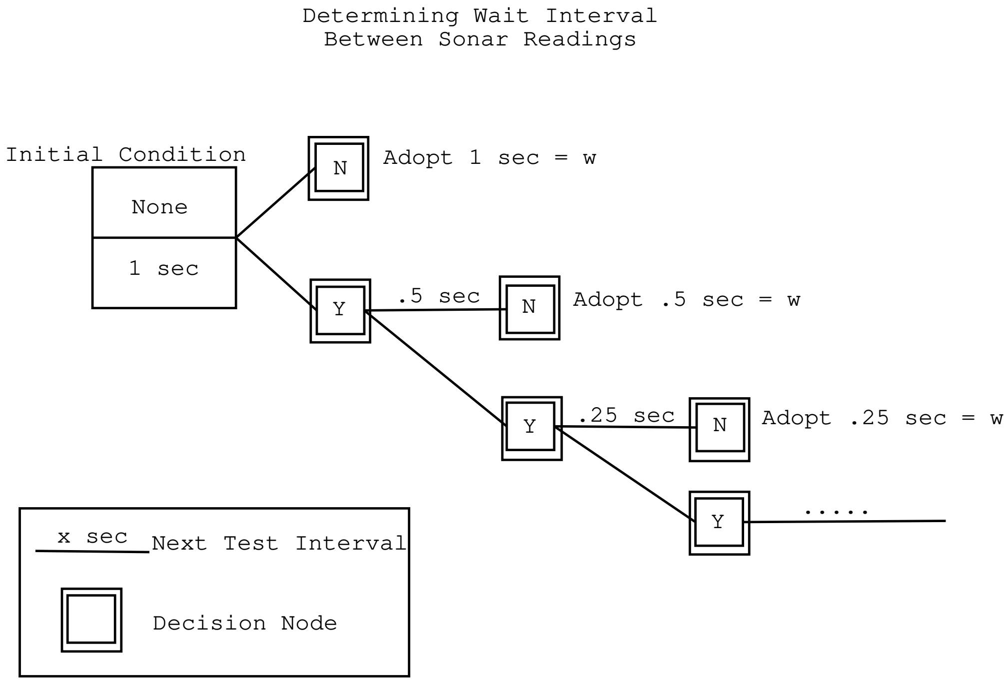

Figure 2: Illustration of experimental process used to determine correct wait interval between sonar readings. The same approach can be used to determine wait between sonar reading and servo movement. The Y or N nodes represent the answer to the question "Is there a significant difference in sonar performance when these two wait intervals are used?"

Experiment 2: Sonar Wait Interval *Variable: Length of "pad" time between sonar readings*

Part a: No waiting vs. 1 second of waiting

1. Set up obstacle within maximum range.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading with no waiting.
4. Sweep servo again from 0 to 180 every 2 degrees.
5. Take a sonar reading with 1 second of sleep before next loop iteration.

If the above experiment yields no difference in sonar readings, skip the next one; otherwise, iteratively determine the appropriate wait time using a "binary search" style strategy for determining hypothetical wait times. Figure 2 is a diagram of the process to be used to determine these wait intervals.

Part b: No waiting vs. 1 second of waiting

1. Set up obstacle within maximum range.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading with no waiting.
4. Sweep servo again from 0 to 180 every 2 degrees.
5. Take a sonar reading with 1 second of sleep before next loop iteration.

Experiment 3: Servo-Sonar Wait Interval *Variable: Length of "pad" time between sonar reading and servo movement*

Part a: No waiting vs. 1 second of waiting

1. Set up obstacle within maximum range.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading with no waiting between sweep and reading.
4. Sleep for amount of time determined in Experiment 2.
5. Sweep servo again from 0 to 180 every 2 degrees.
6. Take a sonar reading with 1 second of sleep before next loop iteration.

If the above experiment yields no difference in sonar readings, skip the next one; otherwise, iteratively determine the appropriate wait time using a "binary search" style strategy for determining hypothetical wait times.

Part b: No waiting vs. 1 second of waiting

1. Set up obstacle within maximum range.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading with no waiting between sweep and reading.
4. Sleep for amount of time determined in Experiment 2.
5. Sweep servo again from 0 to 180 every 2 degrees.
6. Take a sonar reading with 1 second of sleep before next loop iteration.

Experiment Set 4: Obstacle Orientations *Variables: Offset of Obstacle*

Part a: No Offset

1. Set up obstacle within maximum range, perpendicular to center field of view.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Part b: +45 Degree Offset

1. Set up obstacle within maximum range, with its center point at a 45 degree angle away from center field of view.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Part c: -45 Degree Offset

1. Set up obstacle within maximum range, with its center point at a 45 degree angle toward the center field of view.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Experiment Set 5: Obstacle Shapes *Variables: Shape of Object in Field of View*

Part a: Flat Edge

1. Set up obstacle within maximum range to be a flat edge that extends directly in front of the field of view.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Part b: Convex Corner

1. Set up obstacle within maximum range to be a convex corner (i.e., use 2 obstacles angled together) directly in front of the field of view.
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Part c: Concave Corner

1. Set up obstacle within maximum range to be a concave corner (i.e., the corner of one obstacle must be directly in front of the field of view).
2. Sweep servo from 0 to 180 every 2 degrees.
3. Take a sonar reading.

Although there are only 5 experiment sets laid out here, it is important to note that the number of version of Experiment Sets 4 and 5 depend on the results of Experiments 1-3. If it is determined that those variables tested in Exp. 1-3 do not have a significant effect on the accuracy of the sonar, there will be more room for interesting additional experiments, such as those mentioned at the end of last week's document. If they *do* have an effect, then it will be necessary to alter Experiment Sets 4 and 5 to test for the presence and absence of each variable found to have a significant effect initially.

The software template for running sonar-based experiments and collecting data from them has already been implemented by our team in class, and will be the skeleton to which I add the code for the above scenarios.

3.3 Sonar Filter Function

The details of this function will be unknown until after the first phase of the project, the experiments and analysis of their results, is completed. A brief description follows.

The "filter" function will be eventually added to the sonar library: this function will mask from its users the procedure (determined by experimentation) used to obtain the most accurate sonar reading possible. Within the function will be incorporated the points learned in the experiments concerning the validity of the *sonar_closeup()* or *sonar_sample()* methods in certain situations, whether and how to survey the environment, and whether and when to adjust sonar position using the servo motor.

The outputs or computed values of this function will be three quantities: a distance representing how far away the closest point of the closest obstacle is; a length representing the estimated size of the obstacle in the robot's current field of view; and an angle representing the approximate skew of the object away from the robot. These three values will be 0 if there was no obstacle within the detectable range.