Lab 03

Random Wander, Obstacle Avoidance

****

Purpose: An essential characteristic of an autonomous robot is the ability to navigate in an environment safely. The purpose of this lab is to develop random wander and obstacle avoidance behaviors for the Arduino robot. The design of your program should use *subsumption architecture* where layer 0 of your control architecture will be the *collide* and *run away* behaviors to keep the robot from hitting obstacles. Layer 1 will be the *random wander* behavior which moves the robot a random distance and/or heading every n seconds.

Objectives: At the conclusion of this lab, the student should be able to:

- Mount infrared and/or sonar sensors on the Arduino robot
- Write random wander and obstacle avoidance behaviors on the Arduino Robot
- Use modular programming to implement subsumption architecture on the Arduino Robot
- Move the robot safely in an environment with obstacles

Equipment: Arduino Robot

4 Infrared sensors*

4 Ultrasonic sensors (analog and/or digital)*

Tape measurer

**You may need to go check out the sensors from the parts room*

Theory:

Last week, you created the first two primitive motion behaviors on the robot: *Go-To-Goal* and *Go-To-Angle*. It is important to safely navigate a cluttered environment so this week you will implement the *Avoid-Obstacle* and *Random Wander* behaviors. Each week, you will create more functionality and behaviors for your robot so you should design the system to be as modular as possible in order to re-use algorithms and code and build on what you have done before. One way to do this is to use the subsumption architecture with layers where the most basic layer is motion control and obstacle avoidance.

The Sharp IR sensor will return an analog value that is proportional to the distance to an object based upon time of flight for infrared light. The analog ultrasonic sensor emits a sound (ping) and measures the time of flight for the sound to return to calculate distance. The digital ultrasonic sensor uses a trigger pin and echo to perform the same calculation. As with all IR and sonar; the ambient light, color, texture, material, and angle of incidence determine how much energy is returned and this as well as specular reflection may affect accurate measurements.

References:

Arduino Robot Getting Started:	http://arduino.cc/en/Guide/Robot
Arduino Robot Boards:	https://www.arduino.cc/en/Main/Robot
Arduino Language Reference:	http://arduino.cc/en/Reference/HomePage
Arduino Robot Library:	http://arduino.cc/en/Reference/RobotLibrary
Arduino Robot Avoid Obstacle:	http://youtu.be/p1iw-8pXwfI
Sharp IR Sensor Library:	http://playground.arduino.cc/Main/SharpIR
Parallax Ping Sonar:	http://playground.arduino.cc/Code/NewPing https://www.arduino.cc/en/Tutorial/Ping http://playground.arduino.cc/Code/PingInterruptCode
Maxbotix Sonar:	http://playground.arduino.cc/Main/MaxSonar http://www.maxbotix.com/articles/085.htm http://www.instructables.com/id/Getting-started-with-the-Maxbotix-sonar-sensor-q/ http://www.instructables.com/id/Max-Sonar-EZ0/
HC-SR04 Sonar:	http://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/

****

LAB PROCEDURE

****

Part I –Infrared Range Sensor Calibration

1. Mount 1 IR sensor at TK2. Figure 1 shows an image of the Sharp IR sensor.

   Write a program to read the analog value from the sensor by using
   ```
   int value;
   value=Robot.analogRead(SensorPin);
   ```
2. Measure the analog value returned from the sensor for distances from 1” to 20” (see Table 1). Recall that sensors have a range and dead zone so it may not be accurate at certain distances at all.
3. Plot this data in Excel and use curve fitting to find the equation to linearize the data. See an example of how to do this at the following link: https://acroname.com/articles/linearizing-sharp-ranger-data

   Equation: _____
4. You must include a table of the data in your lab memo. You must include a plot of the analog and distance data in your lab memo. You must include the equation found in your lab memo.
5. Modify the program to output inches using the equation you found and create a table of measured versus actual distance for 1” to 20“ with percent error (see Table 1). You must include this table in your lab memo. Please use the following formula for percent error.

$$\%error = \frac{meas - actual}{actual}\ x\ 100$$

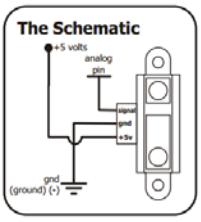


Figure 1: Sharp IR Sensor

Table 1: Sharp IR Sensor Data

Actual Inches	Analog Value	Measured Inches	% error
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

Part II – Sonar Sensor Calibration

1. Attach an analog sonar sensor to TK2 or mount a digital sensor to TKD1. See the "Arduino Robot Pin Table" in the Resources folder on Moodle for the proper way to address the pin in the sketch. If you use the 4-pin digital sensor you should tie the trigger and echo pins together. Figure 2 shows the various sonar sensors.
2. Write a program to read the analog value from the analog sensor by using

   ```
   int value;
   value=Robot.analogRead(SensorPin);
   ```

3. Write a program to read the digital value from the digital sensor by using the following code in a loop.

```
long value;
pinMode(ftSonarPin, OUTPUT);//set the PING pin as an output
Robot.digitalWrite(ftSonarPin, LOW);//set the PING pin low first  delayMicroseconds(2);//wait 2 us
Robot.digitalWrite(ftSonarPin, HIGH);//trigger sonar by a 2 us HIGH PULSE
delayMicroseconds(5);//wait 5 us
Robot.digitalWrite(ftSonarPin, LOW);//set pin low first again
pinMode(ftSonarPin, INPUT);//set pin as input with duration as reception time
value = pulseIn(ftSonarPin, HIGH);//measures how long the pin is high
```

4. Record the data from 1" to 20" in a table similar to Table 1 that must be included in your lab memo.
5. Plot this data in Excel and use curve fitting to find the equation of the line. You must include a plot of the analog/digital and distance data in your lab memo. You must include the equation found in your lab memo. Note that unlike the IR sensor, the sonar sensor data equation should be close to linear.

   Equation: _____

6. Modify the program to output inches using the equation you found and create a table of measured versus actual distance for 1" to 20" with percent error (see Table 1).

a. Maxbotix Analog

b. Parallax Ping Digital

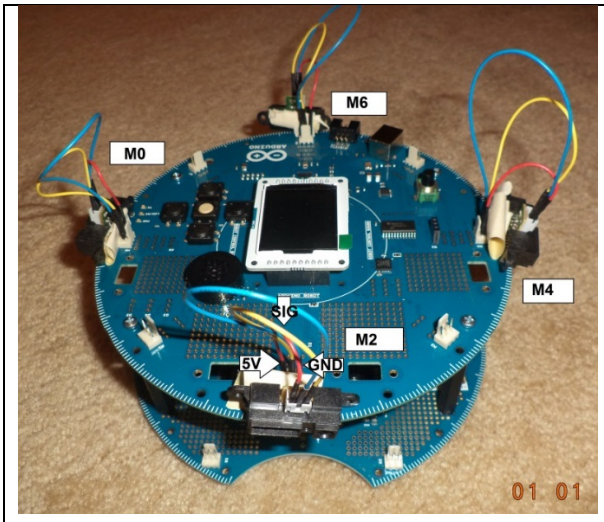
c. HC-SR04 Digital

Figure 2: Sonar Sensors

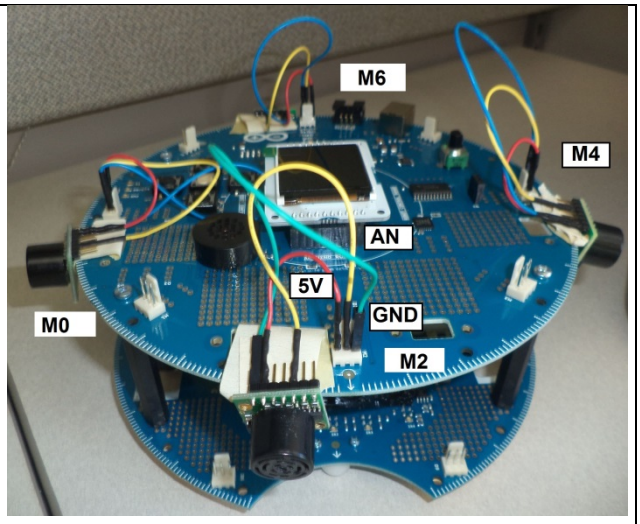
Part III – Mount Sensor Ring

Mount any combination of IR and sonar sensors around the perimeter of your robot. Figure 3 shows two examples with 4 sensors. Note that for a better potential field behavior, you may need to include more than 4 sensors. Mount analog sensors to pins TK0-TK7, TKD0-TKD3, B_TK1-B_TK3 . Mount digital sensors to TKD0-TKD5, B_TK1-B_TK4.

Note: If you don't have enough header pins on the robot and/or sensors, you may have to go to the ECE parts room (C114) to get technicians to solder header pins onto the robot or sensor.



a. Infrared Sensors



a. Sonar Sensors

Figure 3: Sensor Rings

Part IV – Avoid-Obstacle Behavior (Layer 0)

Behavior-based programming uses primitive behaviors as modules for control. Primitive behaviors are concerned with achieving or maintaining a single, time-extended goal. They take inputs from sensors (or other behaviors) and send outputs to actuators (or other behaviors). You will create two more primitive behaviors: *avoid-obstacle*, and *random wander*. The obstacle avoidance behavior will use either *collide* or *run away* behavior. In the collide behavior, the robot will drive forward and stop when an obstacle is detected and continue moving forward when the object is removed. In the run away behavior, the robot will move forward and when an obstacle is detected, move away proportional to where the obstacle is felt (feel force). Your program should be modular and make it clear which behavior is active (collide, run away). The robot's motion will be based upon the potential fields concept.

Now that you are familiar with the range sensors and how to move the robot, create an obstacle avoidance behavior. The *obstacle avoidance* abstract behavior includes a *collide* and *run away* primitive behaviors. For the *collide* behavior, robot would drive forward and if an object is detected within 3 - 6 inches of the range sensors, the robot should halt the forward drive motor. If the object is removed, the robot should continue to move forward.

For the *run away* behavior, create a plot of the sensor readings and use the sum to create a repulsive vector to turn the robot away (i.e. feel force). The robot should turn by some angle proportional to the repulsive vector and continue moving. There are two ways to illustrate the run away behavior, one is to have the robot sit in the middle of the floor and move away when an object gets within 3 to 6 inches and then stop or the robot can start out moving and move away from an obstacle based upon the potential field (see Figure 4).

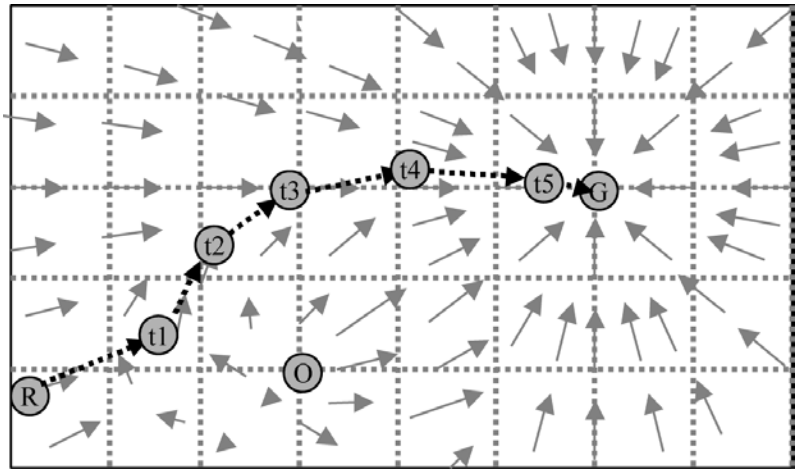


Figure 4: Potential Fields Method for Obstacle Avoidance

To demonstrate the *collide* behavior, the robot should drive forward until it encounters an obstacle and stop without hitting it. Think of the collide behavior as the aggressive kid who comes close but then stops short from touching the object. To demonstrate the *run away* behavior, the robot should sit in the middle of the floor until an object gets close and then move the opposite direction to get away. Think of this behavior as the shy kid who does not want any object to get too close to him. It is possible to also show run away for an aggressive kid who moves forward and then moves away when an object gets close. It should be clear during the demonstration what type of behavior the robot is executing. Figure 5 shows an example of the Avoid-Obstacle behaviors which will be layer 0 of the subsumption architecture.

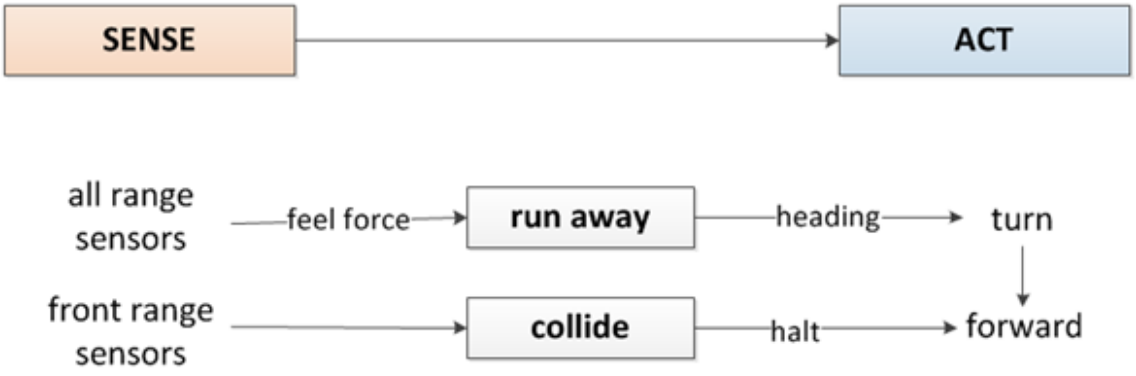


Figure 5: Level 0 – Obstacle Avoidance

Figure 6 provides an example of the robot's sample motion for the Avoid-Obstacle behavior.

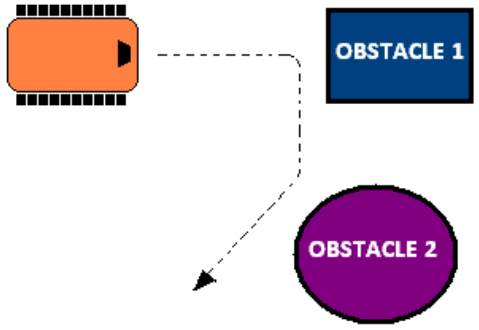


Figure 6: Subsumption Architecture – Sample Robot Motion

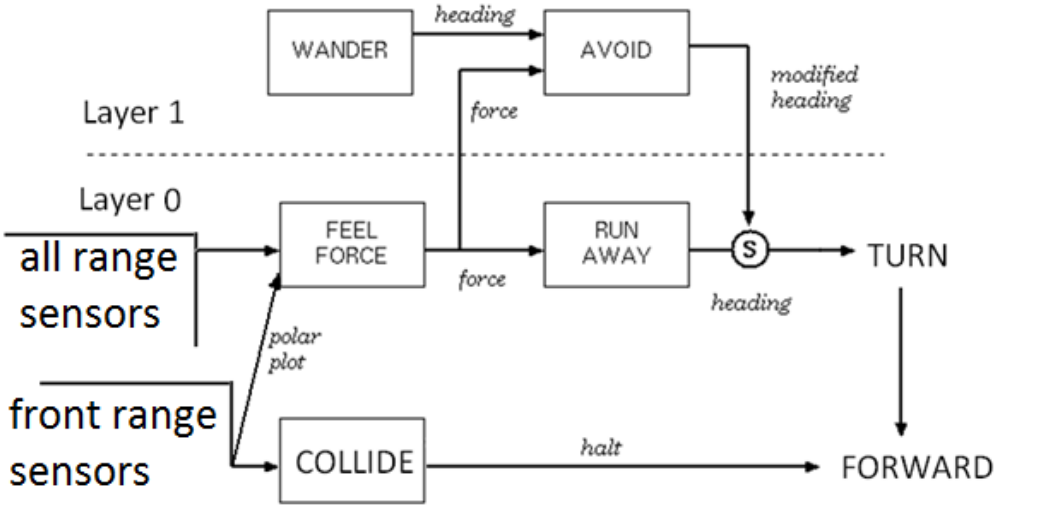
Part V – Random Wander (Layer 1)

In a random wander behavior, the robot will move in a random pattern when no obstacles are present. Create a random wander routine that the robot uses to explore the room. This can be done by generating a random number that represents the robot's heading, distance, or motor speed every n seconds. You have the flexibility of using any combination of these values to make the robot explore the environment. To demonstrate this behavior, set the robot on the floor and execute the random motion.

Part VI – Subsumption Architecture – Smart Wander Behavior (Layer 1)

In this section, you will use the subsumption architecture to create a *smart wander* behavior. This architecture is shown in Figure 7. The perceptual schemas are *feel force* and *collide* and the motor schemas are *run away* and *collide*. The primitive behaviors are *run away* and *collide* and the two together make the abstract behavior, *obstacle avoidance*. The second layer of the architecture is the *wander* module created in part V. The wander module passes the heading the *avoid* module which combines the feel force and wander heading to determine the direction the robot should turn to move away from obstacle. Note that the power of subsumption architecture is that output of the higher level subsumes the output from the lower level. The avoid module suppresses the output from runaway and replaces it to make the robot turn.

Now improve the random wander routine by integrating obstacle avoidance (collide and run away). The robot should wander randomly until an obstacle is encountered. The robot should *run away* from the obstacle and continue to wander. The robot's heading from the *wander* behavior should be modified based upon the force from the range sensors and then turn and move from the obstacle. The *avoid* module in Layer 1 combines the *FeelForce* vector with the *Wander* vector. The *Avoid* module then subsumes the heading from the Run Away module and replaces it with the modified heading as input to the *Turn* module. The power in this type of architecture is the flexibility the execution based upon the use of inhibition and suppression. ***Your code must have the ability to run each of the modules separately as well as together based upon subsuming inputs and inhibiting outputs at different layers.***



s

Figure 7: Smart Wander Behavior

Finally, your program should provide a method to get the robot 'unstuck' if it approaches any local minima points (i.e. oscillates between two obstacles). Your program should be as modular as possible with multiple subroutines and behaviors that will be integrated in subsequent programs. Devise a method to test and confirm that your program works correctly and present the results in the laboratory memo.

Demonstration:

During the demonstration, you will show each layer of the architecture separately.

- For layer 0, the robot should demonstrate shy kid and aggressive kid, separately. Please review the lab procedure if you don't recall what this means.
- For layer 1, to demonstrate random wander, the robot should turn at a random heading and move forward periodically.
- To demonstrate the *Avoid* behavior, the robot should wander in the environment and halt when it "collides" with an obstacle, or modify the heading when it encounters and obstacle and then run away. The robot should give some type of audible and/or visual signal when an obstacle is encountered.

Bring your robot fully charged to class every day!

Program:

In subsequent weeks you will reuse this code thus your code should follow proper programming techniques such as detailed commenting and be as modular as possible where behaviors and reactive rules are separate functions.

Questions to answer in the lab memo

1. Describe the method, pseudocode, flow chart, or state diagram.
2. What was the general plan you used to implement the random wander and obstacle avoidance behaviors?

3. How did you create a modular program and integrate the various layers into the overall program?
4. Did you use the sonar and IR sensors to create redundant sensing?.
5. How could you create a smart wander routine to entirely cover a room?
6. What kind of errors did you encounter with the obstacle avoidance behavior?
7. How could you improve the obstacle avoidance behavior?
8. Were there any obstacles that the robot could not detect?
9. Were there any situations when the range sensors did not give you reliable data?
10. How did you keep track of the robot's states in the program?
11. Did the robot encounter any "stuck" situations? How did you account for those in your code?
12. What should the subsumption architecture look like for the addition of go-to-goal and avoid-obstacle behaviors?

Memo Guidelines:

Please use the following checklist to insure that your memo meets the basic guidelines.

- ✓ Format
  - o Begins with Date, To , From, Subject
  - o Font no larger than 12 point font
  - o Spacing no larger than double space
  - o Written as a combination of sentences or paragraphs and only bulleted list, if necessary
  - o No longer than three pages of text
- ✓ Writing
  - o Memo is organized in a logical order
  - o Writing is direct, concise and to the point
  - o Written in first person from lab partners
  - o Correct grammar, no spelling errors
- ✓ Content
  - o Starts with a statement of purpose
  - o Discusses the strategy or pseudocode for implementing the robot remote control (includes pseudocode, flow chart, state diagram, or control architecture in the appendix)
  - o Discusses the tests and methods performed
  - o States the results and or data tables including error analysis, if required
  - o Shows any required plots or graphs, if required
  - o Answers all questions posed in the lab procedure
  - o Clear statement of conclusions

Grading Rubric:

The lab is worth a total of 30 points and is graded by the following rubric.

Points	Demonstration	Code	Memo
10	Excellent work, the robot performs exactly as required	Properly commented with a header and function comments, easy to follow with modular components	Follows all guidelines and answers all questions posed
7.5	Performs most of the functionality with minor failures	Partial comments and/or not modular with objects	Does not answer some questions and/or has spelling, grammatical, content errors
5	Performs some of the functionality but with major failures or parts missing	No comments, not modular, not easy to follow	Multiple grammatical, format, content, spelling errors, questions not answered
0	Meets none of the design specifications or not submitted	Not submitted	Not submitted

Submission Requirements:

You must submit your properly commented Sketch code & memo to the Moodle DropBox by midnight on Sunday. Check the course calendar for the lab demonstration due date.