Act A

Avoid-Obstacle and Random Wander

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 ROMI 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:

- Acquire and use data from the robot's range sensors
- Write random wander and obstacle avoidance behaviors on the Robot using sensor feedback and a bang-bang or proportional controller
- Use modular programming to implement subsumption architecture or a state machine on the Robot
- Move the robot safely in an environment with obstacles

****

Equipment: Romi base robot

ultrasonic distance sensor (sonar)

3 LEDs (optional)

Theory:

Obstacle avoidance will be based upon sensor feedback. The sonar sensor is used to detect the distance to an obstacle and the error between the desired and actual distance will be used to control the robot wheels. The robot collide behavior will stop the robot if the distance to the obstacle is within a given error band, otherwise the robot continues to move forward. This type of control is BANG-BANG or ON-OFF control because either the controller affects the robot motors or it does not. The robot run away behavior will use the sensor data to create a polar plot and the robot will move away proportional to the distance

to the object as well as the direction of the obstacle. This type of control is called proportional control because the change in robot motion is affected by the magnitude of the error. Figure 1 shows an example of the feedback control system for the robot obstacle avoidance behavior.

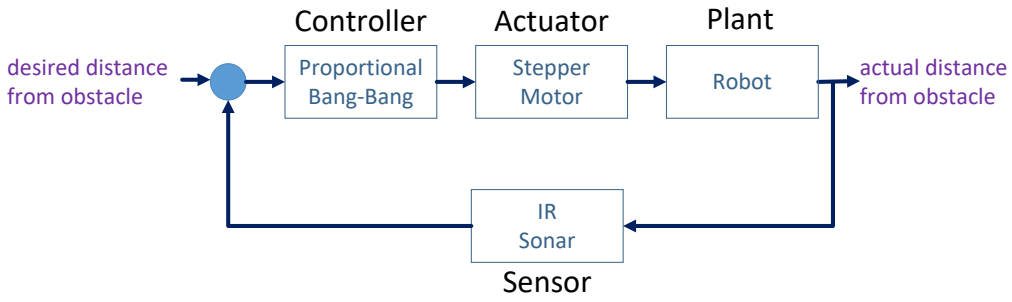


Figure 1: Obstacle Avoidance Feedback Control System

References:

A more efficient way to read sonar data is to use an interrupt that collects distance data in the background while the robot is moving. Also being able to tune a PID controller to adjust robot rate of change or steady-state error when approaching or moving a way from obstacles is a more advanced topic. There are links provided on both of these concepts below.

Timers & Interrupts: http://arduinoinfo.mywikis.net/wiki/Timers-Arduino#Arduino_Timers_and_Interrupts

Interrupts: http://playground.arduino.cc/Code/Interrupts

http://playground.arduino.cc/Code/Timer1

Feedback Systems: An Introduction for Scientists and Engineers,

Karl J. Astrom and Richard M. Murray

http://www.cds.caltech.edu/~murray/amwiki/index.php/Main_Page

Tuning of a PID controller using Ziegler-Nichols Method

https://www.scribd.com/document/233690247/Tuning-of-a-PID-controller-using-Ziegler-Nichols-Method

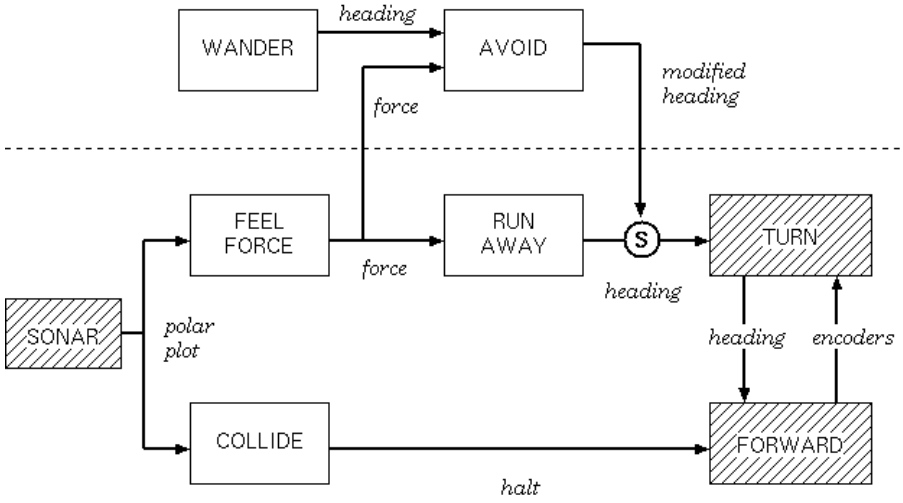
Workshop Note:
Students are required to submit a software design plan at the beginning of the lab assignment. They can create pseudocode, flowchart, subsumption architecture or state diagram. In the lab memo, they must compare & contrast the results to their initial plan.

Pre-Lab:

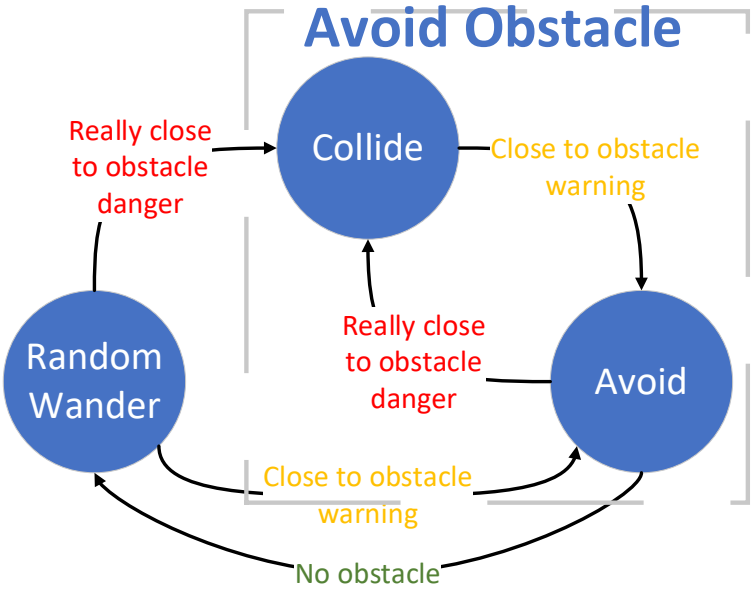
- Create the state diagram and state transition table for the obstacle avoidance behavior with 3 states (random wander, avoid obstacle, go to goal).
- Simulate the collide and runAway behavior in Tinkercad or Dawson Virtual Robot Lab

Solution:
I prefer state diagrams and subsumption architecture for software design plans but will take pseudocode and flowcharts for those who are just starting this level of documentation. By the end of the quarter, I expect them to use state diagrams for their final project.

- Subsumption Architecture



- State Diagram



- State Transition Table

state	input	next state
Avoid (RunAway) Red Danger	Really close to obstacle	Collide
	No obstacles	Random Wander
Random Wander Green	Close to Obstacle	Avoid
	Really Close to Obstacle	Collide
Collide (Halt) Warning Yellow	Really Close to Obstacle	Avoid

- Tinkercad link (good until 1/16/22):

  https://www.tinkercad.com/things/06vatARZex0-ece425-prelab-02-avoid-obstacle/editel?sharecode=hnUuwTfbrRJ4oxIjlwbuSUz6O1B-WcEV1-N1AnQUdm8

****

LAB PROCEDURE

****

Part A.1 – Random Wander (Layer 1)

In this lab, you will create two behaviors: *avoid-obstacle*, and *random wander*. 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. Turn on the GREEN LED when the robot executes the random wander behavior. Look for the TO DO comments in the "actA1 – random.wander" code for places to modify code.

Part A.2 & A.3 – 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). Figure 2 shows the robot primitives for a behavior-based control architecture. Figure 2 is the Avoid-Obstacle behavior and Layer 0 of the

subsumption architecture. The obstacle avoidance behavior is comprised of the *collide* and *run away* primitive behaviors.

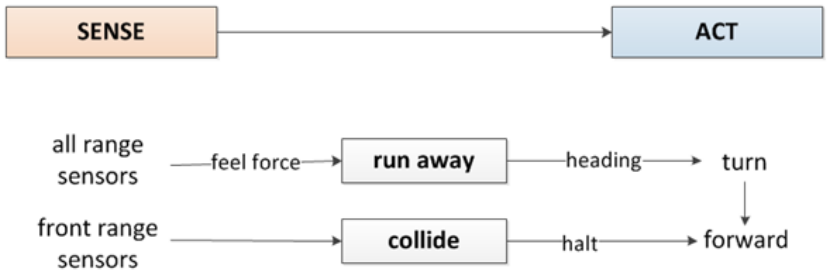


Figure 2: Level 0 – Obstacle Avoidance

Part A.2 – Collide 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. 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. Turn on the RED LED when the robot executes the aggressive kid behavior. Look for the TO DO comments in the "actA2-collide" code for places to modify code.

Part A.3 – RunAway Behavior

In the *runAway* behavior, the robot will move forward and slow down proportional to the distance from the obstacle. This is based upon the robot feeling the force of the obstacle similar to potential field navigation. Alternately, the robot can sit still and when an obstacle is detected, it will move away proportional to where the obstacle is felt (feel force). The robot can also turn by some angle proportional to the repulsive vector and continue moving. 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 based upon the potential field. 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 slows down and moves away when an object gets close. It should be clear during the demonstration what type of behavior the robot is executing. Turn on the YELLOW LED when the robot executes the shy kid behavior. Look for the TO DO comments in the "actA3 – avoid.obstacle" code for places to modify code.

Part A.4 – Subsumption Architecture – Smart Wander Behavior (Layers 0 and 1)

In this section, you will use the subsumption architecture to create a *smart wander* behavior (see Figure 3). The perceptual schema is *feel force* and the motor schemas are *run away* and *collide*. The primitive behaviors are also called *run away* and *collide* and the two together make the abstract behavior, *obstacle avoidance*. The second layer of the architecture is the *wander* module. The wander module passes the heading to the *avoid* behavior 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 also can suppress the output from runaway and replace it to make the robot turn.

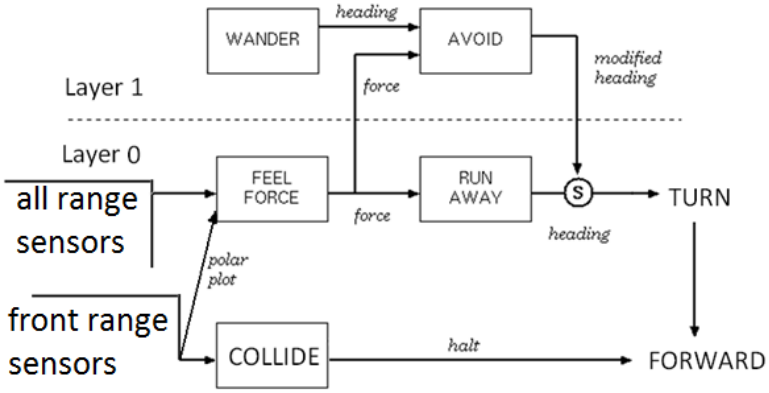


Figure 3: Smart Wander Behavior

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 in the execution based upon the use of inhibition and suppression. Figure 4 provides an example of the robot's sample motion for the Avoid-Obstacle behavior.

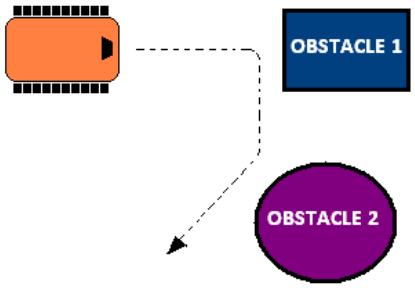


Figure 4: Subsumption Architecture – Sample Robot Motion

Your program should provide a method to get the robot 'unstuck' if it approaches any local minima points (i.e. oscillates between two obstacles). This can be injecting random noise or using a timer to break out of a state after a prescribed period of time. 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. Look for the TO DO comments in the "actA4 – smart.wander" code for places to modify code.

Workshop Note:
Part A.5 is not part of the workshop activities but provide an example of an enhancement to increase difficulty. I teach this course for undergraduate and graduate students and sometimes need two levels of assignments. Students also build on this in the follow on lab when they use PD control for wall following and navigating corners as well as in and out of doors. They also make a state machine to randomly wander into a wall and then start following it.

Part A.5 – Integrate Avoid-Obstacle and Go-To-Goal Behaviors (Layer 2) [Optional]

An alternate environment exploration technique would be to send the robot to a specific goal point or via points. The final step in the procedure will be to integrate the Avoid-Obstacle and Go-To-Goal Behaviors. The robot should attempt to drive to a goal position and stop within a given error. If an obstacle is encountered, the robot should navigate around the obstacle until there is a direct line of sight to the goal position and then continue moving to the goal position. Note that this behavior would have to subsume the *random-wander* behavior.

It is necessary to use potential fields to solve this problem by finding the sum of the goal attraction vector and the obstacle repulsion vector. It is important to keep track of the robot's state and position as it avoids the obstacle in order to calculate the new vector to the goal and also to identify when to abandon the obstacle and make a straight path to goal. This can be done with dead reckoning, encoders or an IMU.

This is also referred to as potential field navigation. Add pushbuttons to the robot to enter a goal location in relative world coordinates. Enter the x- and y-values such as (2', 3') or (12", -18"), where the x-axis is straight out of the front of the robot and the y-axis is out of the left wheel. Figures 5 and 6 are examples of using potential fields to move a robot toward a goal while avoiding obstacles. Figure 7 shows an example of the state machine hat can be used to switch between avoid-obstacle and go-to-goal. Turn on RED, YELLOW, and GREEN LEDs when the robot is moving toward the goal.

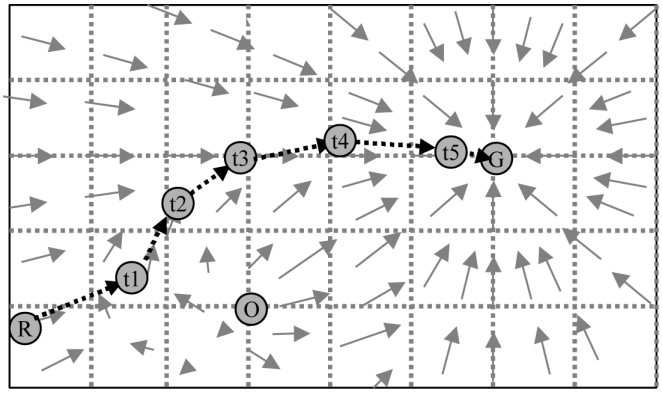


Figure 5: Potential Fields Method for Obstacle Avoidance

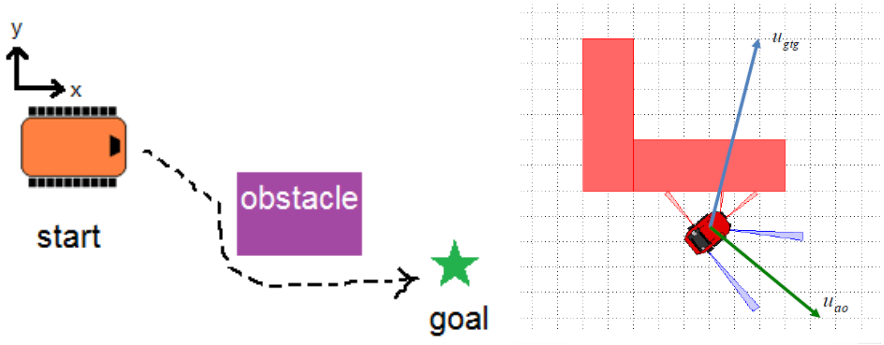


Figure 6: Avoid-Obstacle & Go-To-Goal Behaviors

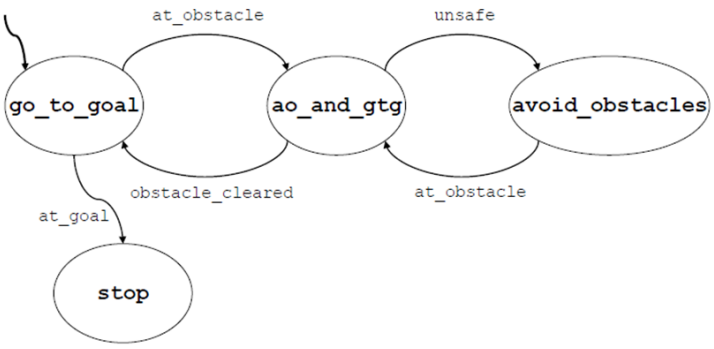


Figure 7: Avoid-Obstacle & Go-To-Goal State Machine

Workshop Note:
Students typically get between one and two weeks to complete a lab assignment. It is dependent upon the level of difficulty. This one was about a week and a half but was over Christmas break. They submit the memo and code the Sunday after they do the lab demonstration in class.

Submission Requirements:

Software Design Plan

For each lab you will submit a software design plan which may be in the form of pseudocode, a flowchart, state machine or subsumption architecture. You will show this plan to the instructor during class, the plan will be graded and you will get feedback on implementation before designing the full system.

Demonstration:

Bring your robot fully charged to class every day! Plug it in overnight.

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. Use RED, GREEN or YELLOW LEDs as described in the lab procedure.
- The final component of the demonstration involves giving the robot a goal position and the robot should move to this location while also avoiding obstacles. The robot should turn on the RED, YELLOW, and GREEN LED when it is moving toward the goal.

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.

Use the following guidelines for creating your code.

- Code has a header comment with
- Program Name
    - Author Names
    - Date Created
    - Overall program description
    - Summary of key functions created with a description of each
- Code is modular with multiple functions called from the loop function. Functions have logical names that describe their functionality.
- Code is organized in a logical fashion with global and local variables with intuitive names.
- Code has block comments to introduce and describe each function and sufficient in line comments to describe how key parts of the code work.
- All functions and variables should have logical names that describe what they do or the value they hold. There should be no magic numbers or numeric values not related to a constant or variable to make it clear what it does.
- In general, someone unfamiliar with your project should be able to read your code and understand the functionality based upon your comments and naming convention.

Memo Guidelines:

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

✓ Format

- Begins with Date, To , From, Subject
- Font no larger than 12 point font
- Spacing no larger than double space
- Includes handwritten initials of both partners at the top of the memo next to the names
- Written as a paragraph not bulleted list

✓ Writing

- Memo is organized in a logical order

- Writing is direct, concise and to the point
- Written in first person from lab partners
- Correct grammar, no spelling errors

✓ Content

- Starts with a statement of purpose
- Discusses the strategy or pseudocode for implementing the robot paths (may include a flow chart)
- Discusses the tests and methods performed
- States the results including error analysis
- Shows data tables with error analysis and required plots or graphs
- Answers all questions posed in the lab procedure
- Clear statement of conclusions
- Include software design plan in the appendix (reference design plan in the text)
- Include screen shot of simulator results or code in the appendix and reference in text

Questions to Answer in the Memo:

1. What was the general plan you used to implement the random wander and obstacle avoidance behaviors?
2. How did you create a modular program and integrate the two layers into the overall program?
3. Did you use the contact and IR sensors to create redundant sensing on the robot's front half.
4. How could you create a smart wander routine to entirely cover a room?
5. What kind of errors did you encounter with the obstacle avoidance behavior?
6. How could you improve the obstacle avoidance behavior?
7. Were there any obstacles that the robot could not detect?
8. Were there any situations when the range sensors did not give you reliable data?
9. How did you keep track of the robot's states in the program?

10. Did the robot encounter any "stuck" situations? How did you account for those?
11. How did you keep track of the goal position and robot states as it integrated avoid-obstacle and go-to-goal behaviors?
12. What should the subsumption architecture look like for the addition of the go-to-goal and avoid-obstacle behaviors?
13. Compare the performance of your robot to the theory and software design plan you made.

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

My Class Website and YouTube Channel:

- https://wordpress.rose-hulman.edu/berry123/mobile-robotics
- https://youtube.com/playlist?list=PL175eO9NwPXKBjluGn5bVpIFitP6FIRav