**Design Proposal for Robotic Wall Following Technique Using Ultrasonic Sensors**

Christopher Hood

Stuart Kant

Matthew McGraw

Anya Skomorokhova

ECE 2031, Digital Design Lab

Section L06

Georgia Institute of Technology

School of Electrical and Computer Engineering

Submitted

November 18, 2011

# Executive Summary

The team will design a SCOMP wall following program for the Amigobot using sonar feedback to determine what velocity commands to send to each wheel, enabling the robot to locate and travel parallel to a wall. The wall has inner and outer corners of 90˚ that the robot must navigate.

The team will design a solution to this problem in the form of a state machine program that executes commands based on the current state and the measured distance to the nearest wall. The state machine consists of six states: “forward motion,” “inside turn,” “outside turn,” “adjust outward,” and “adjust inward.” In “forward motion,” the robot will move forward full speed until it senses a wall in front of it or no wall is sensed. If a wall is sensed in front of the Amigobot, the program switches to the “inside turn” state, turns 90˚, then proceeds to travel forward. If no wall is sensed, the Amigobot will switch to the “outside turn” state, turn 90˚, and then continue forward. A switch is used to toggle between following a wall on the left or on the right. The switch position will determine turning direction, and it will select which ultrasonic sensors will be used to measure distance. Parallel motion to the wall will be maintained by switching to “adjust outward” and “adjust inward” states, which will correct the robot trajectory if the measured distance is not within an acceptable range of 20 ∓ 2 centimeters.

The strength of this approach is that navigation of wall corners will not depend on sensor data. Sensor data is not a dependable measure of distance to the wall when the sensor is not perpendicular to the wall. As the Amigobot turns, the sensors are not aligned with the wall and give incorrect measurements of distance, which can result in collisions with the wall and loss of orientation relative to the wall.

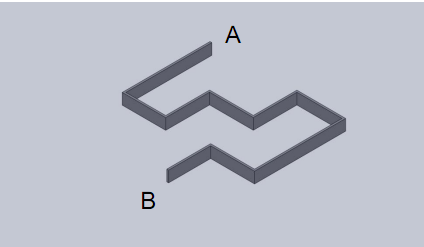
# Technical Approach

## Design Problem

The goal of this project is to design a SCOMP wall following program that will enable the Amigobot to follow walls. The design solution must meet the following specifications:

1. Use an eight bit velocity command where +127 (0x007F) is full speed forward, -127 (-FFFE) is full speed reverse, and zero is stop
2. Control position by reading the cumulative rotation counter of the wheel
3. Use velocity feedback and position feedback from the wheels via the existing optical encoder peripheral
4. Provide a start button to begin execution after the robot is placed adjacent to a wall
5. Use existing sonar and velocity control peripherals to issue commands to each wheel
6. Travel parallel to a wall at a distance of 20 centimeters that has inside and outside corners.
7. Select by switch or recompile to follow left or right walls

The Amigobot is expected to navigate a course without collisions and in a specified time frame. A sample course layout is shown in Figure 1.



**Figure 1.** Sample course layout for robotic wall-following.

In addition to the required specifications, the wall following program will improve the user interface on the robot by enhancing the 7-segment, LCD, and LED displays.

## Design Solution

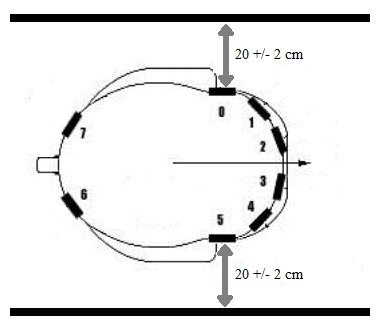
### 2.2.1 Process and Sensor Description

The wall following algorithm to be implemented by the team follows the process found in Figure 2.

**Figure 2.** Flow chart of wall following algorithm.

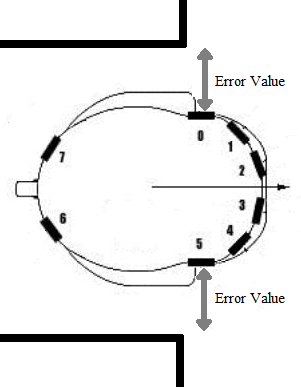
A switch will be used to toggle between following the left wall and following the right wall. The value of the switch determines which sensors are actively collecting data. The values of the following sensors will be used:

* Sensor 0 or 5:
  + Measure the distance to the closest parallel wall (Figure 3)
  + Measure lack of parallel wall by reading maximum value of 0xFFFF (Figure 4)
* Sensor 2 or 3:
  + Measure the distance to approaching wall (Figure 5)

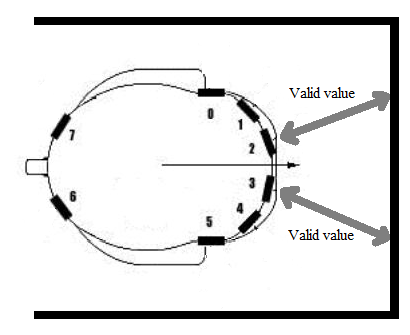


**Figure 3.** Sensors 0 and 5 measuring

distance to wall.



**Figure 4.** Sensors 0 and 5 detecting outside corner.

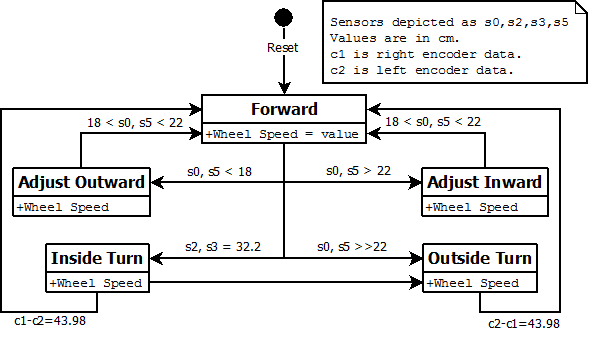


**Figure 5.** Sensors 2 and 3 measuring distance to front wall.

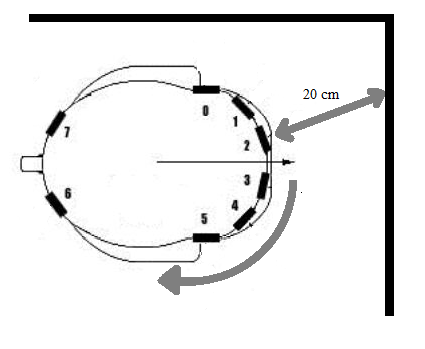
### 2.2.2 State Machine Description

The description of the states is listed below. The UML state machine diagram the team used to implement a solution is found in Figure 6.

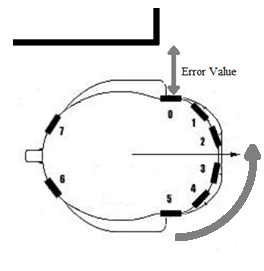
1. Forward: The robot will move forward alongside a wall. A tolerance of ∓2 cm will be used to keep the robot in a range of 18-22 centimeters from the wall, measured by sensors 0 and 5. If the robot is not in the specified range, it will switch to one of the adjustment states, which are dependent on which wall is followed. If a wall is detected in front of the robot by sensors 2 and 3, the robot will switch to the inside turn state. Likewise, if no wall is detected by sensors 0 and 5, the robot will switch to the outside turn state.
2. Adjust Outward: The robot veers slightly outwards to get back within the accepted distance range. After the robot is within the accepted distance range, the machine switches to the forward state.
3. Adjust Inward: The robot veers slightly inwards to get back within the accepted distance range. After the robot is within the accepted distance range, the machine switches to the forward state.
4. Inside Turn: The robot stops and turns 90˚ clockwise or counterclockwise, depending on if the wall followed was on the right side or the left side. Figure 7 demonstrates this turn when the robot is following a left wall and is turning clockwise.
5. Outside Turn: The robot stops and turns 90˚ clockwise or counterclockwise, depending on if the wall followed was on the right side or the left side. Figure 8 demonstrates this turn when the robot is following a left wall and is turning counterclockwise.



**Figure 6.** UML state machine diagram for wall following algorithm for Amigobot.



**Figure 7.** Amigobot making a clockwise turn at an inside wall corner.



**Figure 8.** Amigobot making a counterclockwise turn at outside wall corner.

### 2.2.3 Rotary Encoding

<insert description of turning and math>

<insert diagram of the math of the turning>

<insert facts used to calculate the turn>

## 2.2.4 Amigobot Display Additions

Several additions will be made to the robot display to improve the user interface. Memory locations used to write to and read from are designated in all capital letters.

* Two displays will be used to show the velocity of the left and right wheels.
  + These will be written to SEVENSEG whenever LVELCMD or RVELCMD, the wheel velocities, are changed.
* The third display will show the velocity of the robot.
  + This will be done by altering the IO\_decoder and the BDF files so that the second set of four 7-segment displays can be written to. It will be called SEVENSEG2 and mapped to &H05 on the IO address space map.
  + IO\_decoder will be altered by adding and enabling a signal for the second set of the 7-segment LEDs and copy the setup with the HEX\_DISP module used for the current 7-segment display.
* The LCD display will show the current state. The state name will either be encoded or spelled out on the display
  + This will be done by altering the LCD display to accept ASCII values representing numbers. The SLCD will be altered to take in an ASCII enable, ASCII\_EN, which will tell it to interpret the incoming argument as a state to be output in ASCII format.
  + The states will be encoded into a binary code of 16 bits. For example, “forward” would be encoded as 0x0 and “adjust left” as 0x1. The strings shown on the LCD display will be hard coded using the binary state representations.
* The red LEDs will light up to show the progress of a turn as the robot is turning
  + This will be done by altering the IO\_decoder and the BDF files so that the green LEDs can be used. It will be called RLEDS and mapped to &H05 on the IO address space map.
* The green LEDs will be used to display a pattern unique to the state that the robot is in
  + This will be done by altering the IO\_decoder and the BDF files so that the green LEDs can be used. It will be called GLEDS and mapped to &H05 on the IO address space map.

# Management Plan

The design will be executed according to the Gantt chart provided in Appendix A.

# 3.1 Contingency Plan

In the event that the team cannot complete all tasks before the deadline, the team will implement the state machine with the states “forward,” “adjust inward,” “adjust outward,” “inside turn,” and “outside turn.” These requirements are fundamental to the SCOMP solution algorithm. Additional features outlined in Section 2.2.4 will be implemented as time and resources permit. ADD ON

# 4. Appendix A

