Control of Mobile Robotics

CDA4621.001 S17

Dr. Alfredo Weitzenfeld

Motion Task

Tyler Simoni & Esthevan Romeiro

February 6, 2017

1. Lab Objectives

The objectives for this lab were complete the described tasks listed below using either the motors or IR sensors only. They could not be used in conjunction with any other sensors on the robot. The task execution can be viewed here on YouTube. Each task below also is hyperlinked to the appropriate timestamp in the video.

- IR Sensor – Short Distance
- IR Sensor – Long Distance
- Front – Back open loop
- Circle open loop (CW & CCW)
- Square open loop (CW & CCW)
- Figure-8 open loop

2. Lab Design Overview

2.1. IR Sensor – Short Distance

The robot was take input from each of the short range IR sensors and display the approximate distance, in inches, on the LCD screen.

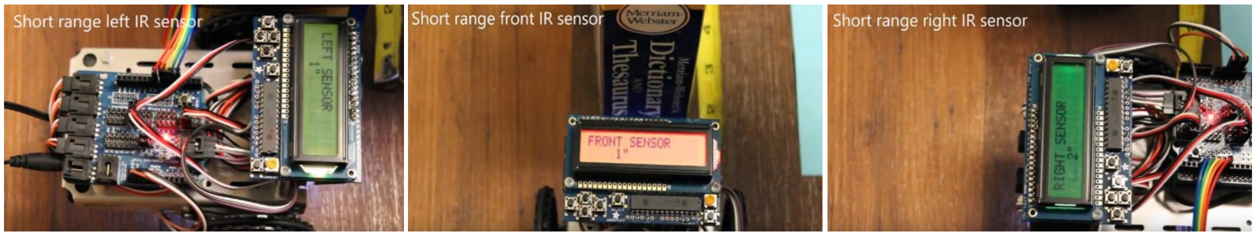
Short range left IR sensor
LEFT SENSOR
1"
Short range front IR sensor
FRONT SENSOR
1"
Short range right IR sensor
RIGHT SENSOR
2"

Figure 2-1. Short sensor readings

In order to do this, the sensor ranges needed to be assigned to their approximate distance in inches. This was done by writing a function to save the min and max values from the `readSensor()` function and display them in the console. The readings were taken at each inch increment up to 10 inches. Using the range for each distance, a `sensorDispay()` function was written, containing a large `if-else if` chain.

Next, an `options` variable was declared and initialized to zero. The setup initializes the LCD screen. When the begins to execute, the robot waits for input from the buttons. The UP button displays "Front Sensor", changes the LCD color, and changes the `options` variable to 1. As long as options is set to 1, the robot will dynamically read and display the front sensor data on the second row of the screen.

The logic works the same for the LEFT and RIGHT button press, where LEFT displays "Left Sensor",sets `options` to 2, and reads the left IR sensor; and RIGHT displays "Right Sensor", sets `options` to 3, and reads the right IR sensor.

2.2. IR Sensor – Long Distance

The robot was to take input from the long range IR sensor in the front and display the approximate distance, in inches, to the LCD screen.

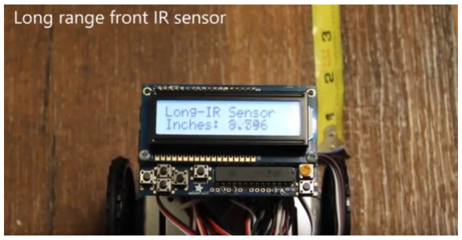

Figure 2-2. Long sensor readings

To accomplish this, the sensor readings needed to be associated with the distance. Using the `analogRead()` function, the average of 1000 voltage readings from the IR sensor were taken at 1 inch increments from 10 inches to 31 inches away. This data was input into an Excel spreadsheet and a graph of distance vs. voltage was generated. An exponential trend line with an equation was created from this data.

DISTANCE	VOLTAGE
10	465.87
11	439.42
12	405.05
13	378.58
14	350.78
15	327.03
16	300.75
17	283.91
18	268.11
19	255.41
20	238.68
21	228.74
22	218.37
23	211.71
24	202.93
25	196.31
26	190.52
27	182.62
28	177.27
29	170.48
30	163.96
31	157.72

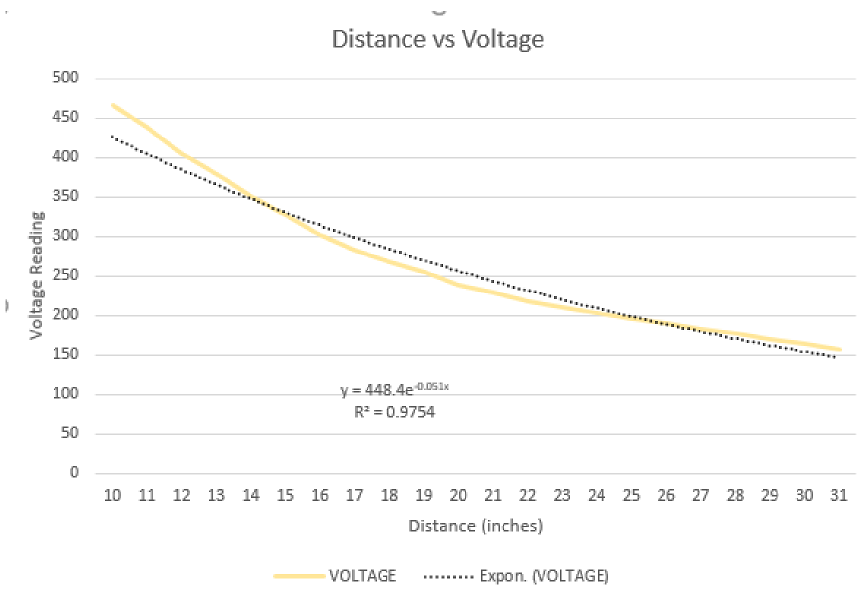

Figure 2-3. Data points and graph for long IR sensor

This type of trend line was chosen as it most closely resembled the input data. The equation was then solved for *x* and implemented in the Arduino script, where x is `inches` and y is the voltage reading (`sensorValue`) from the sensor. This value was then displayed on the LCD screen every loop iteration.

2.3. Front – Back Open Loop

The robot was to drive between two marks, 70 inches apart. First going forward, as straight as possible, then going backward once it reached the other mark.

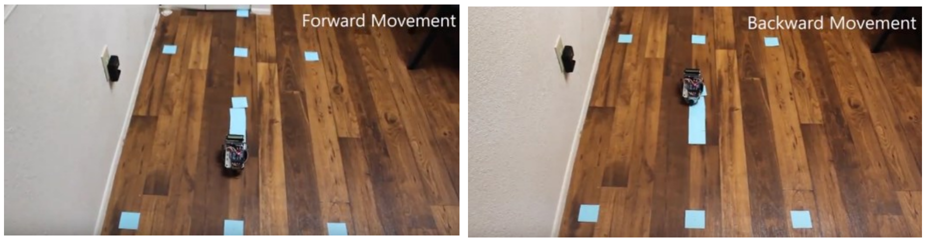

Figure 2-4. Forward and Backward movement

To do this, we set the speed of the wheels and then measured out 70 inches. Because we could not use other sensors, we timed about how long it took to drive that distance. A delay timer was then implemented in the code for that time in both directions.

2.4. Circle Open Loop (CW & CCW)

The robot was to drive in a circle with a radius of 15 inches, in both clockwise and counter-clockwise directions.

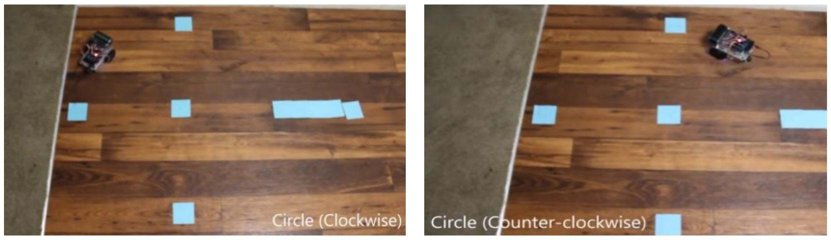

Figure 2-5. CW & CCW circles

First, a loop was measured and drawn out on the floor. We then played with the speed settings on the servos to get an approximate circle in one direction. After some fine tuning with the speed settings, the circle path was quite close to the measured circle. In order to do a single circle and then stop, we measured the time it took and implemented that time as a delay in the loop.

The circle in the other direction was done similarly; however, as the servos are slightly different in operation, a little more tuning went into implementing it.

2.5. Square Open Loop (CW & CCW)

The robot was to drive in a square with each side measuring 30 inches, in clockwise and counter-clockwise paths. The lab description showed the robot starting in the middle of one of the sides, so the loop code was adapted to do this.

Figure 2-6. CW & CCW squares

First, we timed how long the robot took to drive approximately 30 inches in a straight line. This was then divided by two and set as a delay. The loop then drives straight for that delay, stops for a second, turns approximately 90 degrees, and then drives straight for the above mentioned delay. This was then set it in a loop iterating 4 times so that a start delay could be implemented before each square path was driven. This was done for both the clockwise and counter-clockwise paths.

2.6. Figure – 8 Open Loop

The robot was to drive in figure-8 type motion where each circle of the figure-8 was 30 inches and circles were 15 inches apart. The robot was to start on the edge of the outside circle as per the diagram in the lab handout.

Figure 2-7. Figure-8 motion

For this part, the robot drives in two S-bend paths, so that once it completes both, it ends in approximately the same location as before. These S-bend paths were implemented as separate functions (`sLBend` and `sRBend`) and called by the main `loop()` function after a 10 second delay to allow for setup.

3. Conclusions

There were many things in this lab that could have been improved upon. For the most part, the issues could have been solved with better hardware. With better and more expensive hardware, comes more accurate and consistent results.

The first issue, came from the short range IR sensors. With both the robot and the target standing perfectly still, the sensor reading fluctuates anywhere from 20 to 40 units. This required ranges to be set in order to output an accurate distance measurement.

Next was the long range IR sensor. The distance for this one was calculated using the `analogRead()` function which reads the voltage of the sensor. The voltage reading used to generate the formula was an average of 1000 iterations. Also, there was a lot of noise created from all the different electronic modules, which caused the sensor reading value to fluctuate and, in turn, caused the inches reading to fluctuate. This caused the long range IR sensor to not read as accurately as it could.

The rest of the motion tasks all suffered from the same few problems. The main issue was caused from the battery discharge. As the robot was tested and timed to get more and more accurate, the batteries obviously will discharge. As the batteries discharge, the servos become less and less responsive, which means they go slower and turn response gets worse. This makes it hard to test for consistency. This could be solved with either a better battery pack or a more constant voltage source.

The other issue that came up was our two servos functioned differently. The same values for turns in opposite directions would yield very different turn radii.

All in all, our group was successful in the completion of this lab, albeit took significantly more time to test and debug than it did to actually write the bulk amount of the code.