CSCE236 pROJECT 2

SPRING 2016

May 1, 2016

eNDI XU, Nguyen LUU, JEREMY Fitz

Blue Dagger

Contents

[**Introduction** 2](#_Toc450128275)

[**Project Overview** 2](#_Toc450128276)

[**Robot Name and Exterior** 2](#_Toc450128277)

[**System Architecture** 3](#_Toc450128278)

[**Hardware** 3](#_Toc450128279)

[**Light Sensor** 3](#_Toc450128280)

[**Range Finder** 3](#_Toc450128281)

[**Servo Motors** 4](#_Toc450128282)

[**Switch and Led** 4](#_Toc450128283)

[**Software** 5](#_Toc450128284)

[**Function Description** 5](#_Toc450128285)

[**Main Loop Construction** 5](#_Toc450128286)

[**Range Finder and Sensor Characterization** 6](#_Toc450128287)

[**Range Finder** 6](#_Toc450128288)

[**Light Sensor** 8](#_Toc450128289)

[**Approach** 8](#_Toc450128290)

[**Check Point 1: Wall Following** 8](#_Toc450128291)

[**Check Point 2: Obstacle Avoidance** 9](#_Toc450128292)

[**Final Competition** 10](#_Toc450128293)

[**Performance Evaluation** 11](#_Toc450128294)

[**Successes and Failures** 11](#_Toc450128295)

[**Improvements which could be made** 11](#_Toc450128296)

[**Conclusion** 11](#_Toc450128297)

# **Introduction**

## **Project Overview**

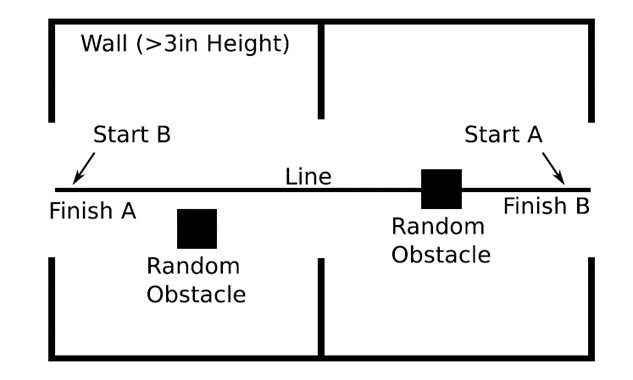
For this project, a robot is designed to compete against others in a modified capture the flag-type environment with two exits. The robot will start at one exit and the goal will be to get out of the opposite exit before the opponent. Figure 1 shows an overview of the environment. There will be a black line on the ground connecting the two exits which can be followed to the exit. The walls surround the environment is at least 3 inches high. Also, two random obstacles will be placed in it.

Figure 1 Overview of the Environment

In this report, the system architecture of the robot, range finder and sensor characterization, approach used and performance evaluation for the competition will be discussed.

## **Robot Name and Exterior**

The robot’s name is Blue Dagger. The robot has a blue base and can fight against its opponent like a dagger. This is why it is called “Blue Dagger”.

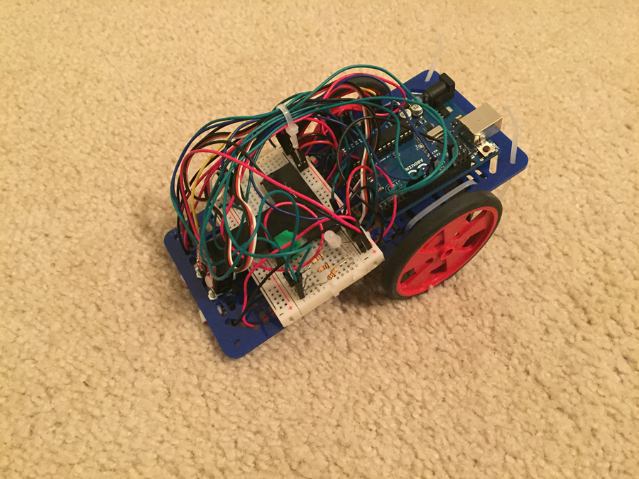
 Figure 2 shows the exterior of the robot.

Figure 2 Robot Exterior

# **System Architecture**

## **Hardware**

### **Light Sensor**

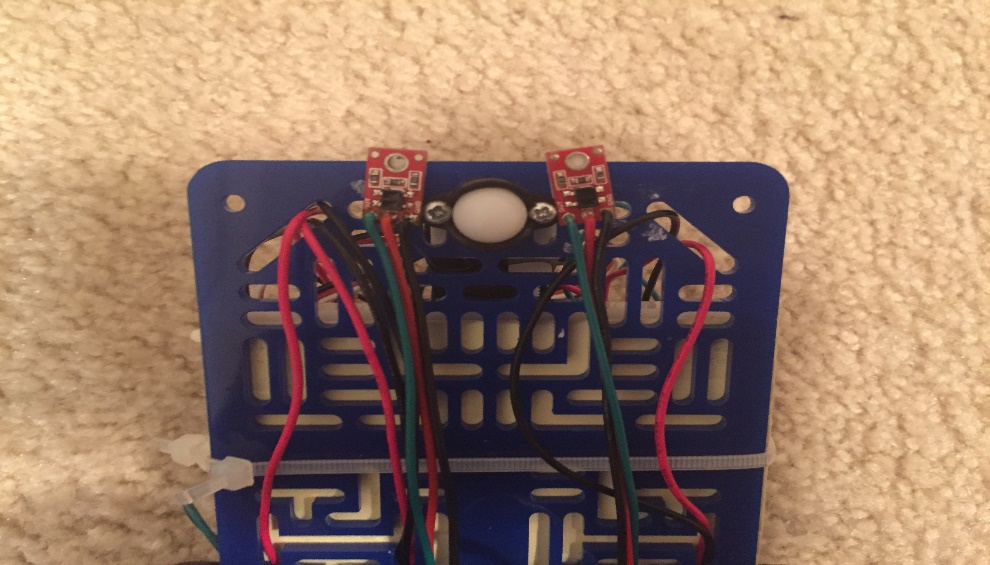
 Figure 3 shows the position of the two light sensors.

Figure 3 Position of the light sensor

As shown in the figure 3, the two sensors are mounted at the two sides of the nose wheel. The approach used for this robot is straddling the line, which means the robot will keep one sensor on the line and one off the line (The details will be discussed in the Approach section).

The reason of mounting the sensors at the sides of the nose wheel is that the sensors can obtain the data before the body of the robot arrives so the robot can have enough time to do the corresponding action (go straight, turn left or turn right) based on the data obtained by the sensors. Also, in this way, it is easy for people to detect whether one sensor is on the line and another sensor is off the line, which is easy for debug.

### **Range Finder**

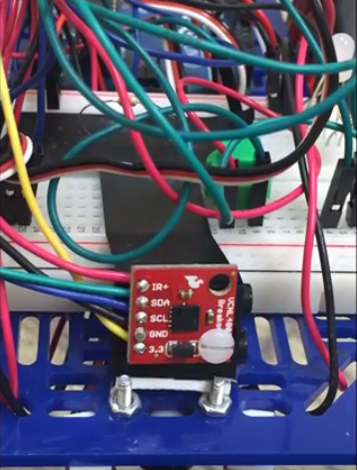
Figure 4 shows the position of the range finder, which is in front of the robot.

Figure 4 Position of the range finder

The reason to put the range find in front of the robot is that throught this way, the robot can detect whether there is an obsacle ahead directly through the reading of the range finder. Only the obstacle ahead needs to be detected. The obstacle located at position can be ignored since it will not hinder the robot.

### **Servo Motors**

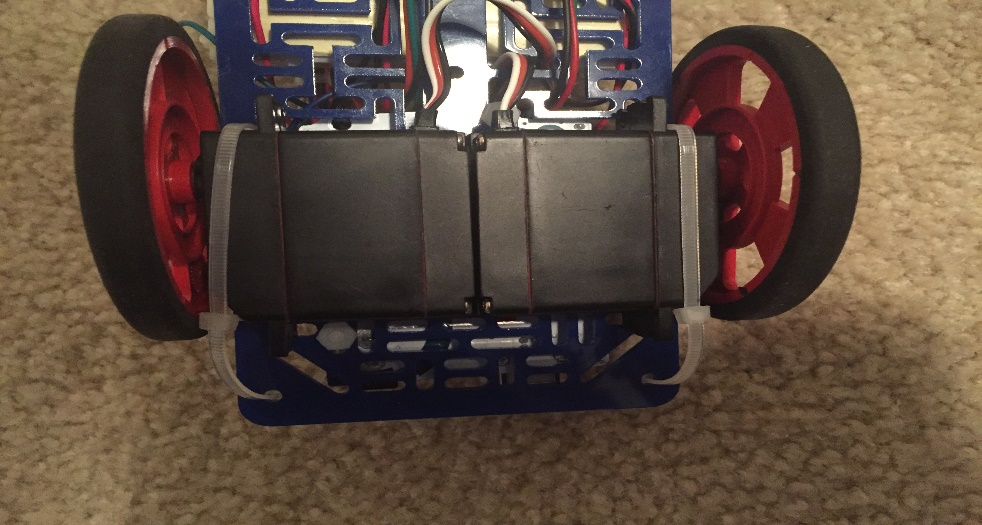
 Figure 5 shows the position of the two servo motors.

Figure 5 Position of the Servo Motors

The two servo motors is used to control the two back wheels separately. Through this way, the speed of the two back wheels can be adjusted independently to make the robot turn right, turn left, go straight or go reverse.

### **Switch and Led**

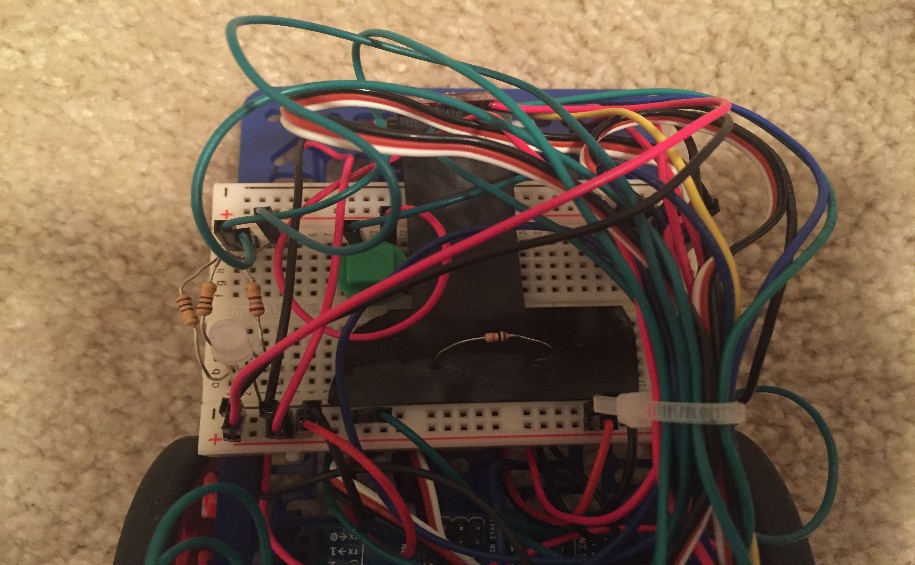
Switch is used to turn on and off the robot. Led is used to indicate the current state of the robot (go straight = green, turn left = red, turn right = blue, reverse = purple). Figure 6 shows the position of them.

Figure 6 Position of the Switch and LED

## **Software**

### **Function Description**

There are total 15 functions used in the final code. They can be divided into 3 groups.

The first group has 3 functions, which are used to get the reading value from the ranging finder. Table 1 shows the details.

|  |  |
| --- | --- |
| Function Name | Description |
| unsigned int readProximity(); | Get Proximity Value |
| void writeByte(byte address, byte data); | writes a single byte of data to address |
| byte readByte(byte address); | reads a single byte of data from address |

Table 1

The second group has 2 functions, which are used to attach the servo and detach the servo. They are used to turn on and off the robot, which are controlled by the switch. Table 2 shows the details.

|  |  |
| --- | --- |
| Function Name | Description |
| void servoAttach(); | Attach the servos |
| void servoDettach(); | Detach the servos |

Table 2

The third group has 10 functions, which refers to the 5 different states (go straight, turn left, turn right, reverse and dodge right) and their corresponding LED lights. Table 3 shows the details.

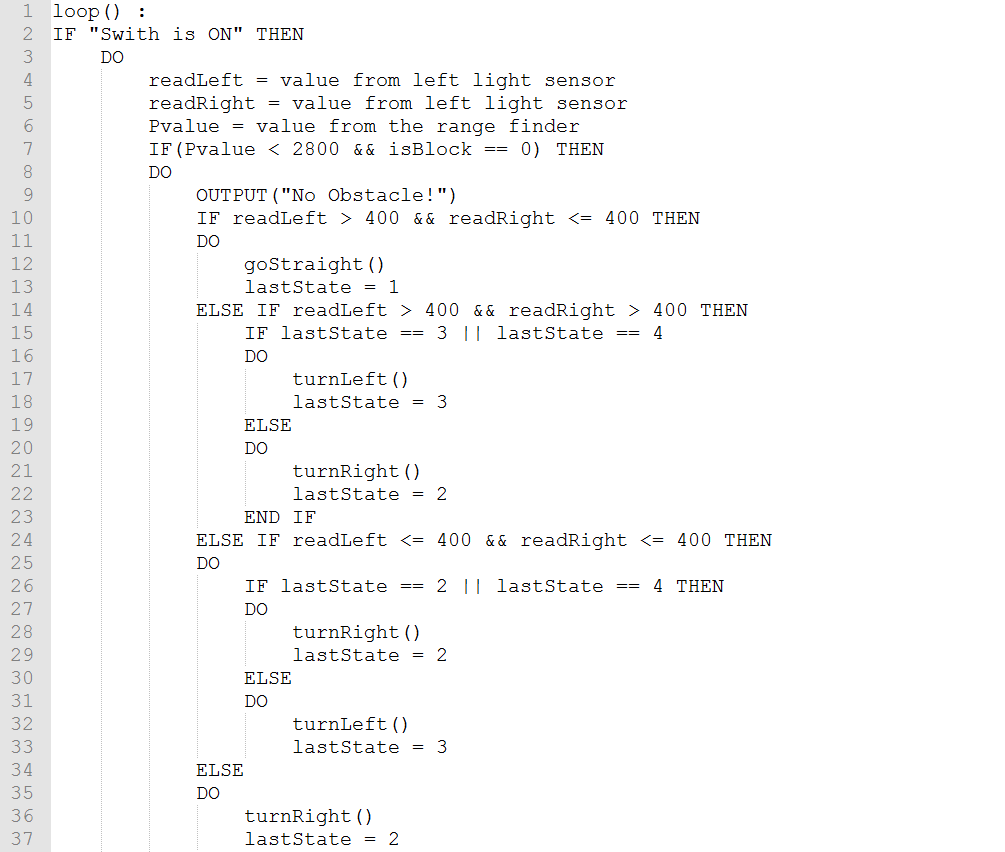
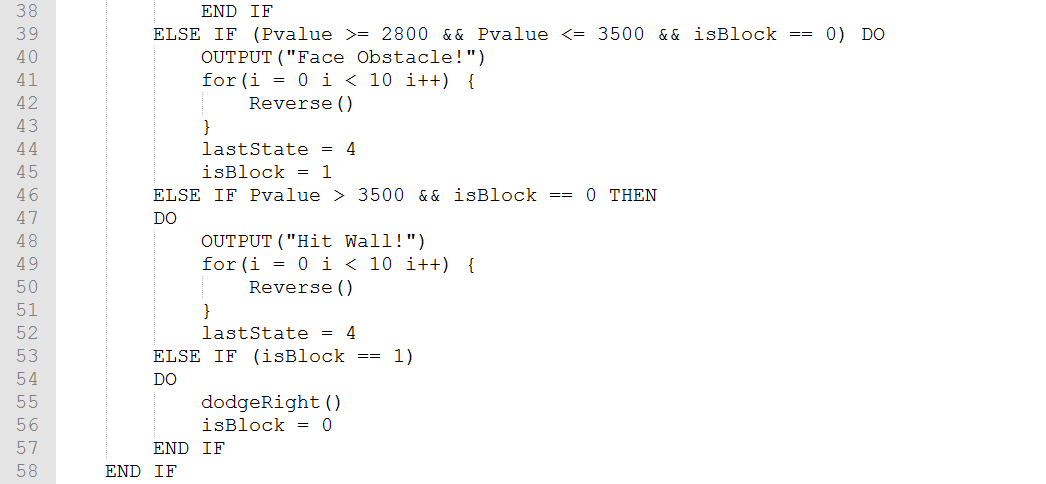
|  |  |
| --- | --- |
| Function Name | Corresponding LED |
| void goStraight(); | ledG(); //Green |
| void turnLeft(); | ledR(); //Red |
| void turn right(); | ledB(); //Blue |
| void reverse(); | ledX(); //Purple |
| void dodge(); | ledB(); ledG(); //Blue then Red |
| Robot stops | ledOFF(); //OFF |

Table 3

### **Main Loop Construction**

In the main loop, all the value read from sensors will be updated first. Then based on the value form range finder, there are 3 conditions, which are “no obstacle”, “face obstacle” and “hit wall”. If the condition is “no obstacle”, just follow the line. If it is “face obstacle”, reverse first and then dodge right. If it is “hit wall”, then go reverse. For following the black line, the robot will move based on the value reading from the two light sensors. The details of it are also introduced in the approach part.

Figure 7 shows the pseudocode of the main loop.

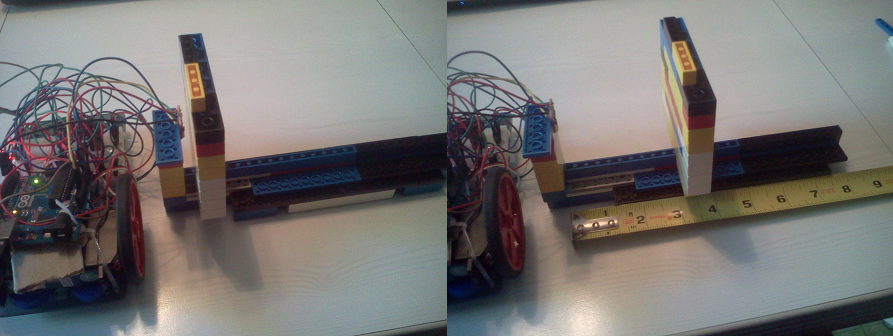
Figure 7 Pseudocode for the Main Loop

# **Range Finder and Sensor Characterization**

## **Range Finder**

The range finder used is a VCNL4000 rangefinder with an ambient light sensor. The sensor is interface with the Arduino using I2C protocol.

To characterize our proximity sensor our group constructed a platform out of LEGO bricks which has a mounting point for the sensor and a movable wall as shown in figure 8.

Figure 8 LEGO Device

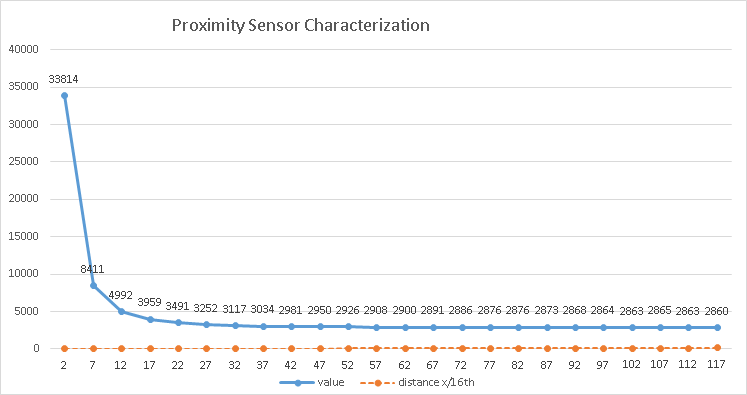
This platform allowed for consistent readings from 1/8” up to 7 - 5/16” in 5/16” increments. The values read from the proximity sensor were taken as the average of 50 readings for each data point. The results show that the readings are not linear. This can be seen in figure 9.

Figure 9 Proximity Sensor: Value vs Distance

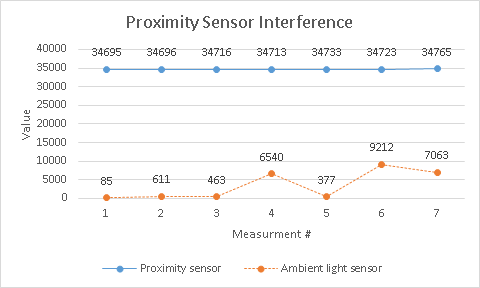
The proximity sensor was also tested for interference from ambient light. Keeping the wall stationary and shining a flashlight at different angles did not seem to affect the proximity sensor significantly. The ambient light sensor however clearly changes values with different lighting conditions as can be seen in figure 10.

Figure 10 Proximity Sensor and Ambient Light Sensor

Due to the fact that the proximity sensor cannot accurately give readings for distance from an object, our group decided to implement the proximity sensor as a sort of “bump switch”. This allowed our robot to poll the sensor and compare the value with a set threshold value to check for when an object was directly in front of the robot. The robot would then execute the appropriate functions to maneuver around it. Using the sensor in this way prevented our robot from needing to come into contact with an obstacle or other robot (most of the time).

## **Light Sensor**

The light sensors interface on the analog bus of our Arduino. The readings are analog voltages which are sent through an analog to digital converter which results in an integer value. This value is used to determine if the sensor is over a light or dark surface.

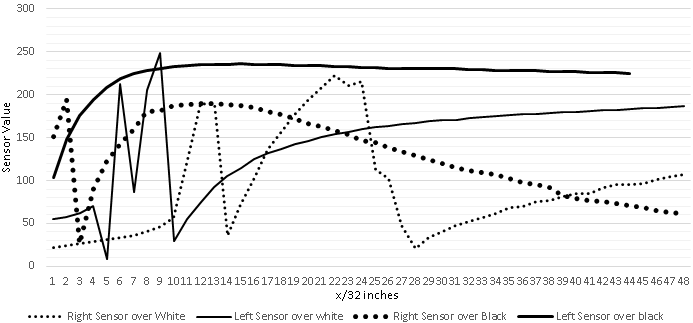
The average reading for our sensors over a black line is a value of 500. The average reading when over a white surface is 300. The threshold value that was used during the checkoffs and the competition was 400. Anything less than 400 was considered on the white and anything over 400 was considered on the black line. Figure 11 shows the reading of the left and right sensor on white and black line respectively.

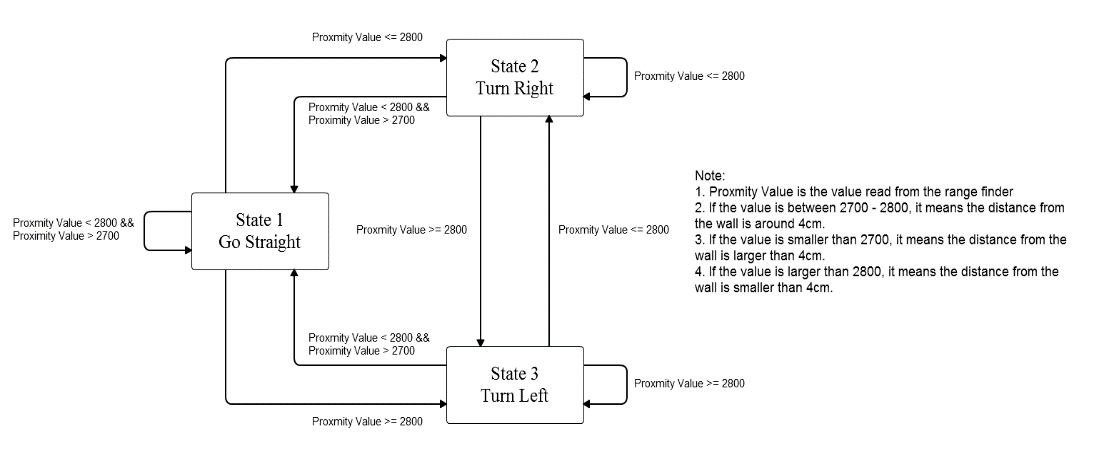
Figure 11 Sensor Value vs Distance

# **Approach**

## **Check Point 1: Wall Following**

In check point 1, the robot is designed to follow a wall for 1 meter through the range finder. For this checkpoint, the range finder should be put at the right side of the robot to detect the distance from the wall. The approach used is simple.

There are three basic states (go straight, turn right and turn left). Based on the value read from the range finder (The value represents the distance from the wall), if the robot is too close to the wall, then move away from the wall (turn left). If the robot is too far to the wall, then move close to the wall (turn right). Otherwise, just move ahead (go straight). Figure 12 shows the state machine diagram of the approach.

Figure 12 State Machine Diagram of Wall Follower

## **Check Point 2: Obstacle Avoidance**

In check point 2, the robot is designed to following a black line and avoid potential obstacle on the black line. Instead of putting the range finder on the right side, the range finder will be put in the front of the robot to detect the obstacle ahead. Also, the two light sensors need to be used as shown in figure 3.

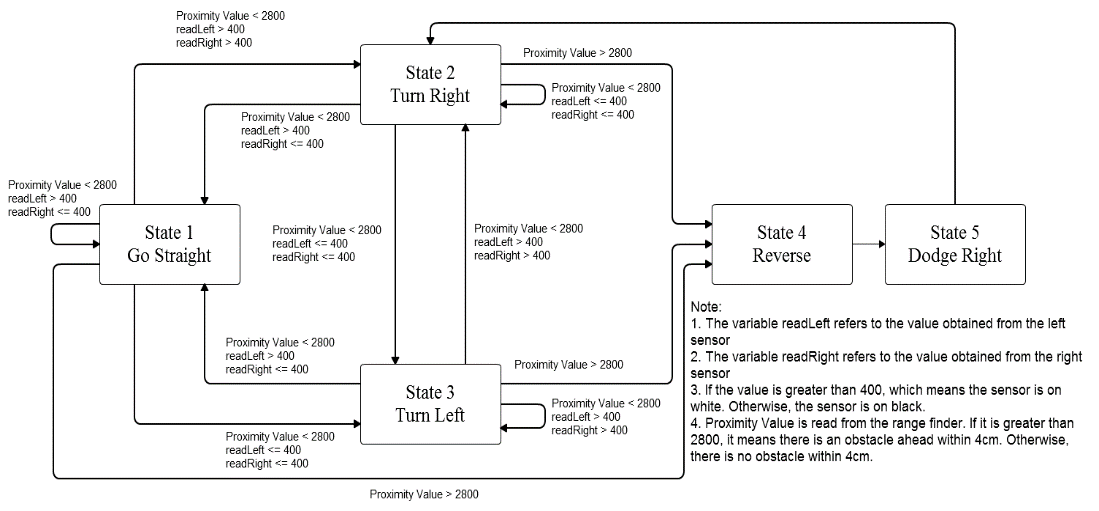
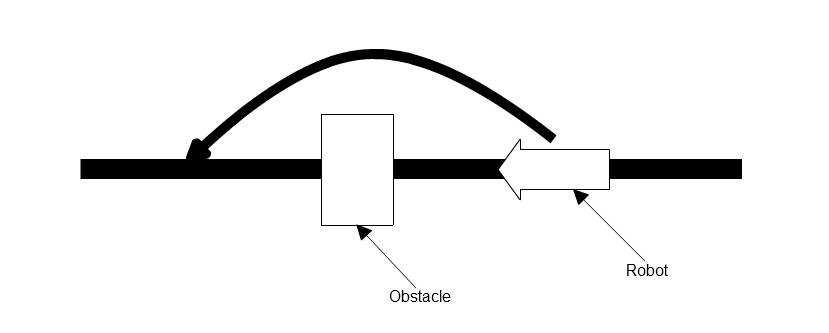
For the approach, two more states which are “reverse” and “dodge right” are added. First, the value of the range finder will be read to check if there is an obstacle ahead or not. If the value is smaller than 2800, it means there is not obstacle ahead. The robot will simply follow the black line. (“Straddle the line” is the approach used to follow the line. The left sensor is always off the line and the right sensor is always on the line. Based on the value read from the two light sensors, the robot will adjust its position to follow the line.) If the value is greater than 2800, it means the robot faces an obstacle. Then it will reverse and dodge right. The robot will go back for a short distance (about 5cm) and avoid the obstacle from its right side. Figure 13 shows the state diagram of the approach and figure 14 shows the path of the dodge right function.

Figure 13 State Machine Diagram of Wall Follower

Figure 14 Path of “Dodge Right”

## **Final Competition**

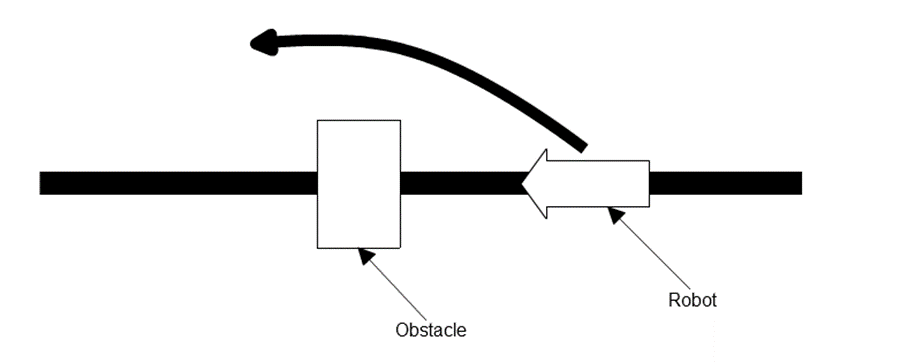
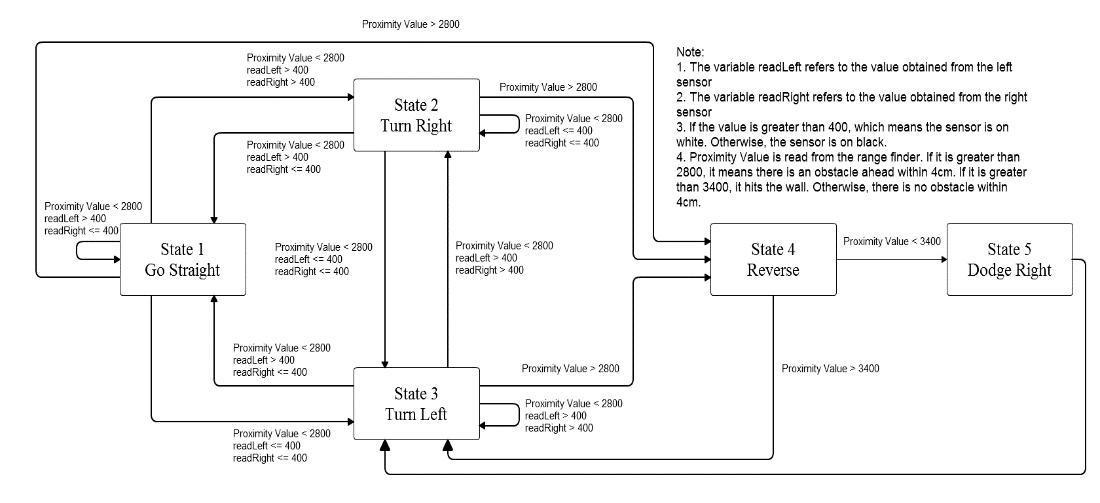
As discussed in the project overview, the final competition is to compete against others in an environment as shown in figure 1. The goal is to get out of the opposite exit before the opponent. Because the robot will also follow a black line and avoid obstacle in front of it, the code used in the final competition is based on the one used in check point 2. However, there is wall surrounding the environment and the position of the obstacle is random. The situation that the robot hits the wall should be considered. In this situation, the robot will go reverse first and keep turning left until it finds the black line. Also, the “dodge right” function is updated, instead of the way shown in figure 6, it will only make a right turn and go straight a short distance. Then, it will keep turn left until find the black line. Figure 15 shows the updated path of the dodge right function and Figure 16 shows the state diagram of the approach.

Figure 15 Updated Path of “Dodge Right”

Figure 16 State Machine Diagram of the Final Competition

# **Performance Evaluation**

Our team totally took three competitions. Our robot won one competition out of the three. In general, the performance is not as well as expected.

## **Successes and Failures**

Without the interfering from other robots or the brightness of the room, the robot can pass the course shown in figure 1 successfully. Also, the speed of the robot is as fast as expected. This is the success of the robot.

The biggest failure is that the robot sometimes turn right when it is supposed to turn left. This is because the reading from the light sensor is unstable. Sometimes, the reading value will be affected by the shadow and the brightness of the room. This will make the robot go to the wrong state and go out of control.

Another failure is that when the robot is hit by opponent’s robot, the robot sometimes will turn over. This is because the robot is too light and the center of gravity is too high.

## **Improvements which could be made**

Because the reading from the light sensor is unstable, a filter function can be made to filter the reading value which is jumped. This can solve the problem that the robot goes to the wrong state due to the unstable reading.

Also, the robot should be made heavier with a lower center of gravity. A shield can be added in front of the robot. This can add the weight of the robot. At the same time, it can help to prevent hitting from other robots.

# **Conclusion**

For this project, the robot can pass the check point 1 and check point 2 successfully. However, during the competition, because of the movement of the opponent and the position of the obstacles are unpredictable, the performance of the robot becomes unstable. This is what the robot needs to be improved.

Through this report, the knowledge of Arduino programming, basic robot construction and I2C protocol are learned.