e-Yantra Robotics Competition - 2014

Implementation Analysis – Cargo Alignment Robot

<Team ID>

Team leader name	**SYED MOHAMMED YOUSUFF HUSSAIN**
College	**National Institute of Technology, Karnataka, Surathkal.**
Email	**yousuff145@gmail.com**
Date	**26-01-2015**

<This report contains three sections:
1. Preparing the Arena
2. Design Analysis
3. Algorithm Analysis
Teams have to answer question/s from these sections according to their understanding of the theme and the given Firebird V robot. >

Preparing the Arena **(5)**

<Prepare the arena according to the steps given in Section 3: Arena, of the rulebook. Take a photo of the complete arena such that the entire arena is clearly visible in the photo. Insert the image here>

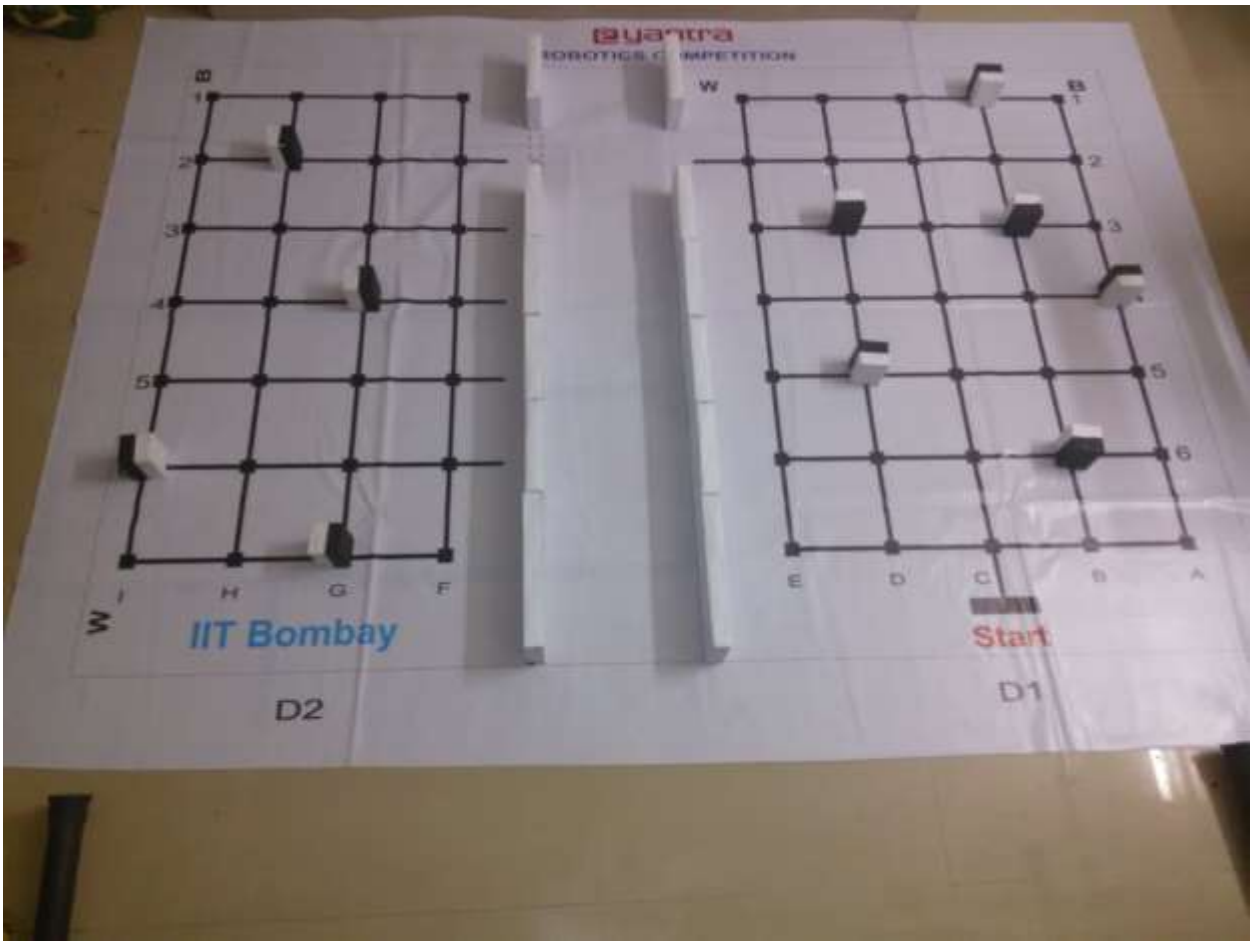


Fig.0

Design Analysis

Q-1. How will you detect the correct/incorrect position of blocks? (10)

<List the sensor(s) you would use to detect the block. Give a brief description of the working of the sensors>

The IR sharp sensor will be used to detect the presence of the blocks and the extra white line sensors will be used to detect the alignment of the blocks.

The white line sensor module given to us consists of three pairs of IR LED's and IR sensors. IR LED emits IR light. This light reflects back from a surface and is received by the IR sensor. The IR sensor is a light dependent resistor, whose resistance changes with the intensity of light. The IR sensor is wired in series with a 10K resistor and the voltage across the sensor is taken as the sensor output.

When more light is incident on the IR sensor, the resistance of the sensor reduces and the voltage drop across it also reduces. This is the case when a white surface is present in front of the sensor.

Similarly, in the absence of light, the sensor resistance is high compared to 10K resistance in series and the voltage drop across the sensor increases. This is the case when a black surface is present in front of the sensor which absorbs all the light and no reflected light falls on the sensor.

Thus, the voltage output is proportional to the intensity of the light falling on the IR sensor. This way, we can determine if the surface in front of the sensor is white or black.

In division D1, we use the front proximity sensor to detect the alignment of the blocks. This sensor also works on the same principle discussed above. Since the bot will approach the blocks always from the front in the current navigation scheme, the front sensor will face either the black or the white surface of the block ahead and we can take that reading to determine if the block ahead is aligned properly or not.

In division D2, the bot will approach the blocks from the side. In this case, we will use the extra white line sensors to detect the alignment of the block. The sensors will be placed on the arm as shown in the figure 1 below. The white line sensors placed on the arm will either be facing the black or the white surface. So, the bot can easily detect the alignment of the cubes.

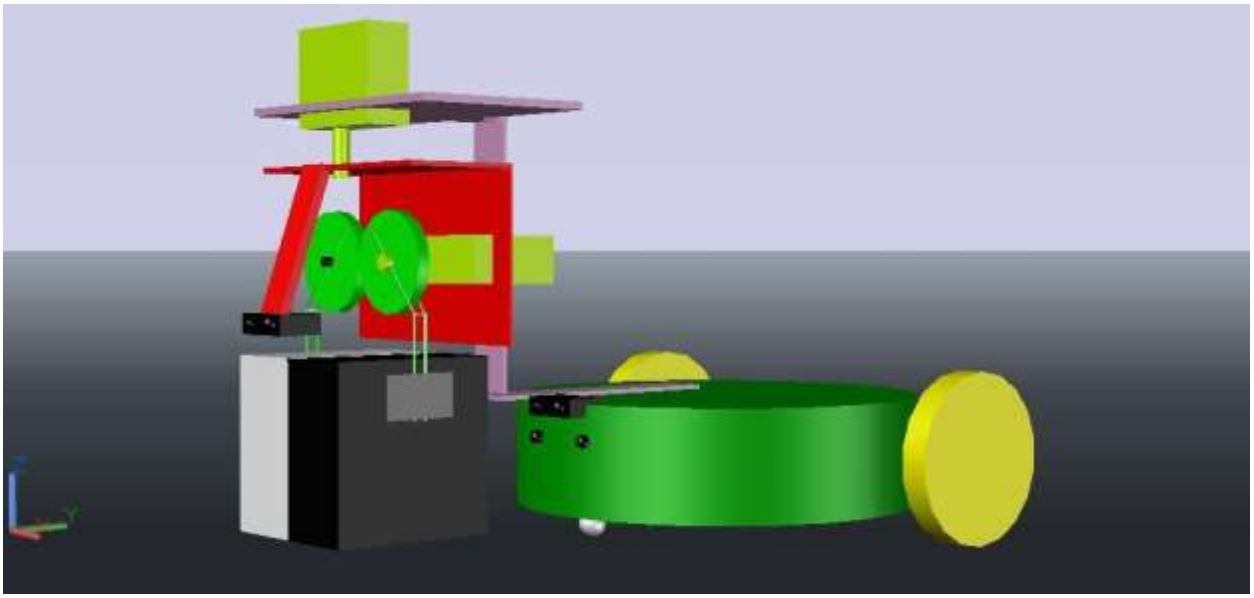


Fig 1

Q-2. Draw a labeled diagram to explain how you have planned to place the sensors on/around the robot? (10)

<Draw a neat diagram to show the positions of sensors on and around the robot. Show all the sensors you are using in designing the theme. Justify placement of the sensors shown in your diagram>

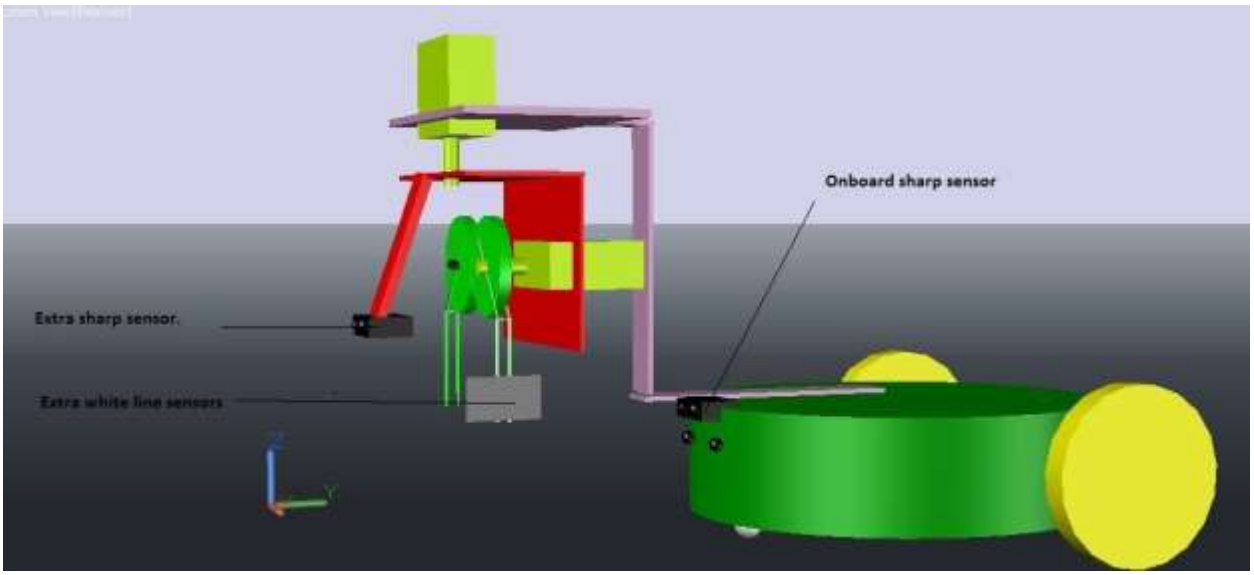


Fig 2

The eight proximity sensors, the white line following sensors below and the sharp sensor onboard are placed in their default positions.

1. The front proximity sensor will be used to detect the alignment of the cube in division D1.
2. The onboard sharp sensor will be used to detect the presence of the cubes ahead.
3. The white line sensors placed on the arm will be used to detect the alignment of the cubes in division D2. When the cube is gripped, the sensor will be facing either the black or the white surface of the cube and the bot will know the exact alignment of the cube.
4. The extra sharp sensor placed on the arm will be used in for two reasons-
   a. To detect the presence of a cube behind another cube in division D1. Refer to fig 2.2. By placing this sensor, we can easily detect the cube behind another cube without having to lift the cube.
   b. To detect the presence of the gate into D2. The gripper will be turned towards wall-2 while the bot will be following wall-1. This way, the sharp sensor can detect whether the gate is present on not.

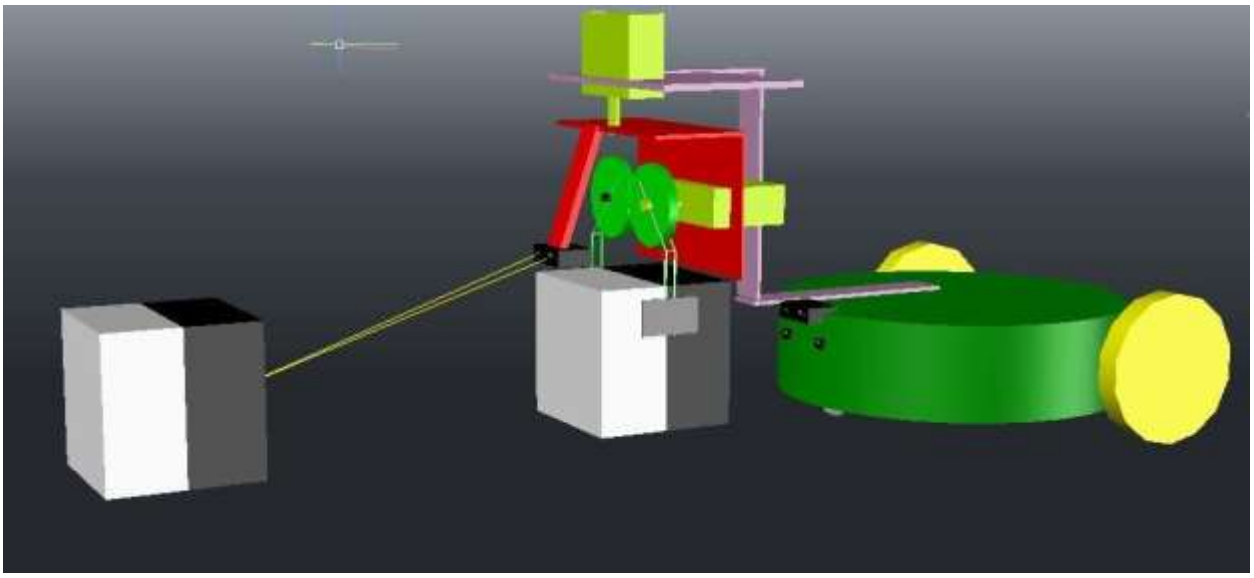


Fig 3

Q-3. Teams have to make the robotic arm for picking up/rotating the blocks in the arena.

a) Choose an option you would like to use to position the robotic arm on the robot and why? (5)

1. Front 2. Back 3. Right/Left 4. On both sides

Answer: Front

< Justify your choice for placement of the robotic arm.
Word-limit: 300 words
>

1. The navigation scheme we developed required the robot to approach the blocks from the front.

2. Keeping the arm on either left or the right side would complicate the wall following algorithm.

3. Another disadvantage we anticipated in keeping the arm on any one side of the bot is that the weight of the bot on that side would increase. This may put more load on the motor on that side and the robot would tend to move towards that side.

4. The arm being in the front will be advantageous in detecting the entry gate into D2. Since we have planned to place the extra sharp sensor on the arm, we can detect the entry gate beforehand. The arm will be turned facing wall-2 while the bot is following wall-1. As soon as the bot detects the entry gate into D2, it will take a right turn and follow its path into D2.

5. To simplify the overall design, we decided to keep the arm in the front.

b) Draw a diagram to show the robotic arm and how it is mounted on the robot. Also show the mounting of the white line sensor. (10)

<Draw figure(s) to show how you are planning to mount the robotic arm and white line sensor on the robot. >

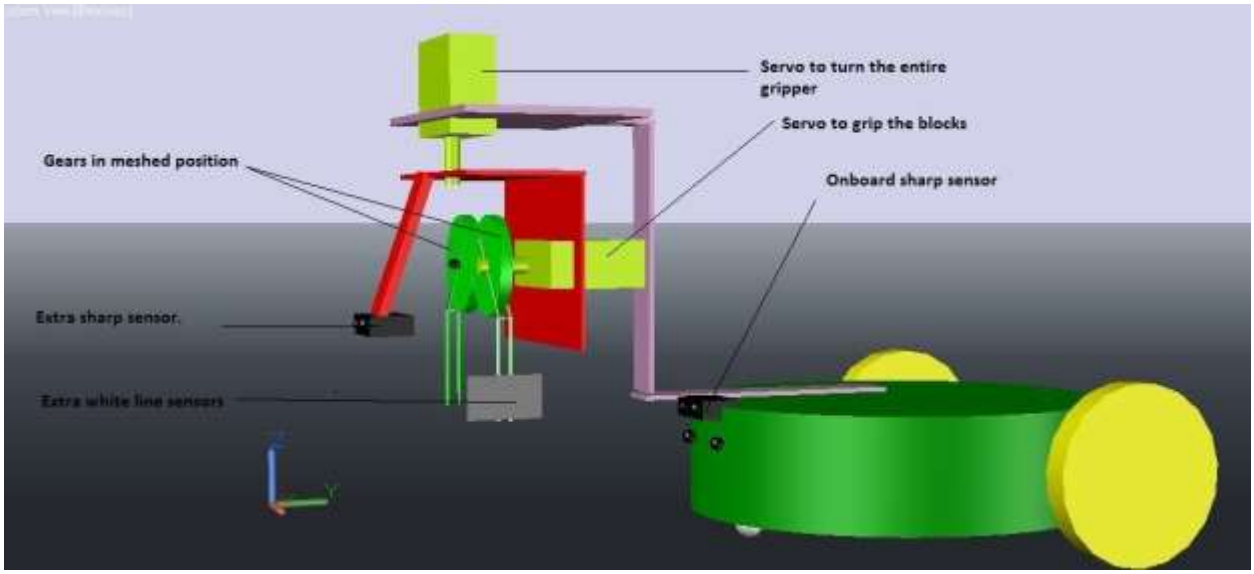


Fig 4

Q-4. Choose the actuator you will use to design the robotic arm. (5)

1. DC-Motor 2. Servo Motor 3. Stepper Motor 4. Others

Answer: Servo motor

Justify your answer by stating the advantage of the chosen actuator over others. Also give reasons for not using other actuators.

1. Servo motors give good precision of control and are easy to control.
2. Servo motors are most usually used for robotics arms as they work with lower voltages and can be turned to required angles with ease.
3. With a DC motor, it is hard to control the degrees of rotation and we may have to use optical encoders to do so, which would complicate the overall design and increase the cost.
4. Stepper motors also give good precision but they are slightly difficult to control and we may require additional drivers for the stepper motors. Also, stepper motors are costlier than servos.
5. We were provided servo motors in the robotic kit

Q-5. How will you detect the block positions in Division 1 and Division 2? Which sensor/s are required to detect the block positions? Explain your algorithm to detect the blocks in both divisions. (15)

<
Answer format: Bulleted form
Step 1:
Step 2:
Step 3…etc.
>

Division D1:

1. In Division D1, we will be using the sharp sensor on board and also the extra sharp sensor to detect the block positions. The bot will keep column C as the reference. At each row, the bot will turn left and right, and check if the there are any cubes present in that row using the onboard sharp sensor.

2. This way, we can successfully check the cubes present on column B or D. We can also detect the cubes present on columns A and E if there are no cubes on column B and D respectively.

3. In case, if there is a cube behind another cube (ex . Cube on (6, A) and (6, B)) , then we cannot detect the cube on column A (or column E) because the cube on column B (or column D) will block the line of sight of the sensor. So, we use the extra sharp sensor to detect the presence of the cube on column A or E. The extra sharp sensor will be mounted as shown in the figure below.

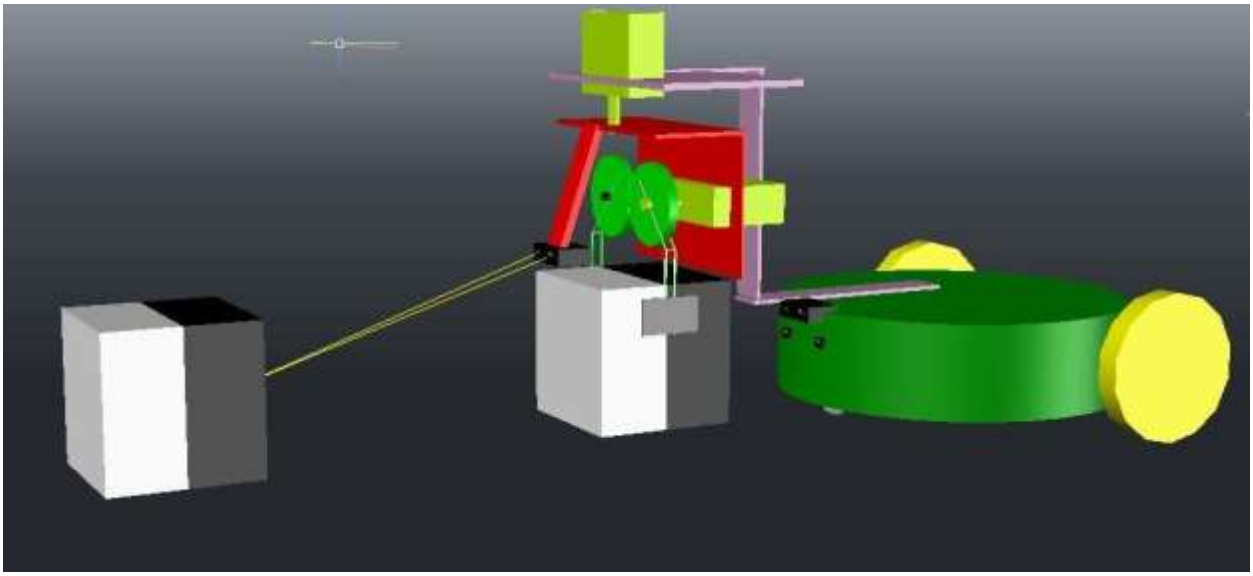


Fig 5.To illustrate the detection of one cube behind another

Division D2:

1. In Division D2, the sharp sensor on board will be used to detect the position of the blocks. Here, a brute force method will be applied. As soon as the bot enters into Division D2, it will move along column F to row 6.

2. The bot will scan each row for a block using the onboard IR sharp sensor. Once the sharp sensor detects the block, the bot will detect it's alignment as explained in Question-1 and align the block accordingly. Then the bot will return to column F, move on to the next row and repeat the same until row 1 is reached.

Q-6. **(15)**

a. How will you connect the white line sensor? Show the schematic pin connection diagram.

<Draw figure(s) to show how you are planning to connect the white line sensor on the robot. >
Answer format:
Step 1:
Step 2:
Step 3…etc.
>

Refer to the schematic below (Zoom in to see the text). The view in the schematic is as shown in the image.

1. The ADC channels 14, 15 (From servo pod) and 9 (sharp sensor 1) are used to read the values from the sensors.
2. The 5 V is taken from one of the unused sharp sensor sockets on the bot.
3. 5K potentiometers will be used to adjust the power to the individual IR LED's for better calibration. A separate general purpose PCB is used to solder the potentiometers. Refer to image 6 below.
4. We are shorting the grounds of IR LED's and the sensors. So, the IR LED's cannot be turned off by software. We will try to add this feature in the future to save the power by turning off the IR LED's when they are not needed.

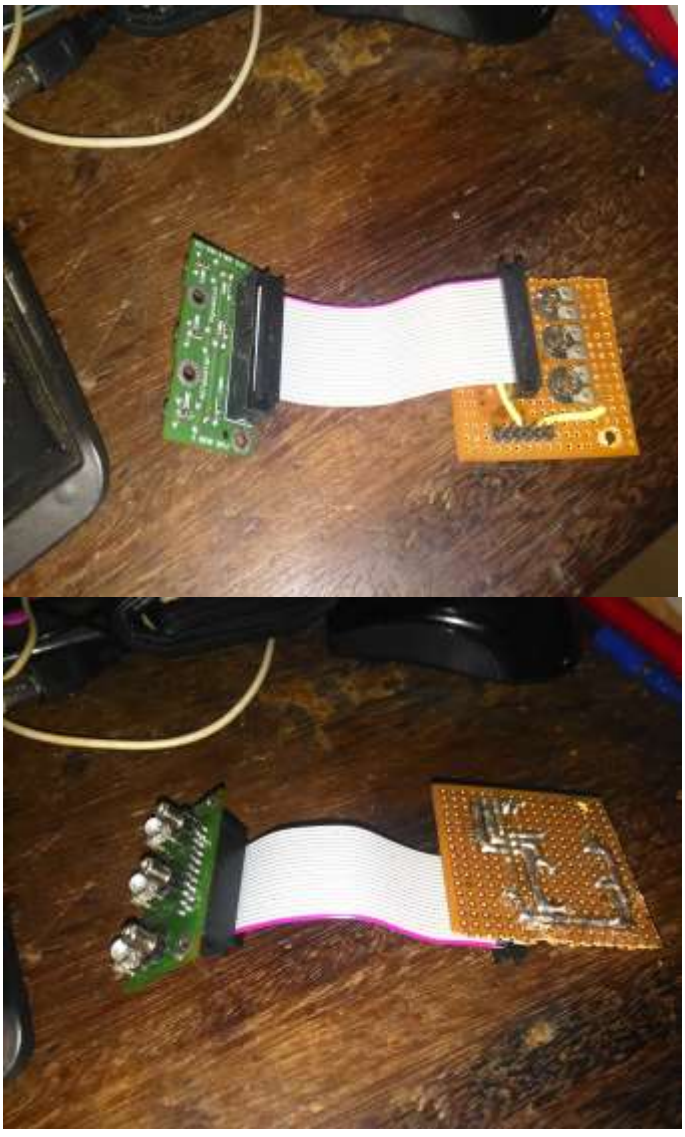
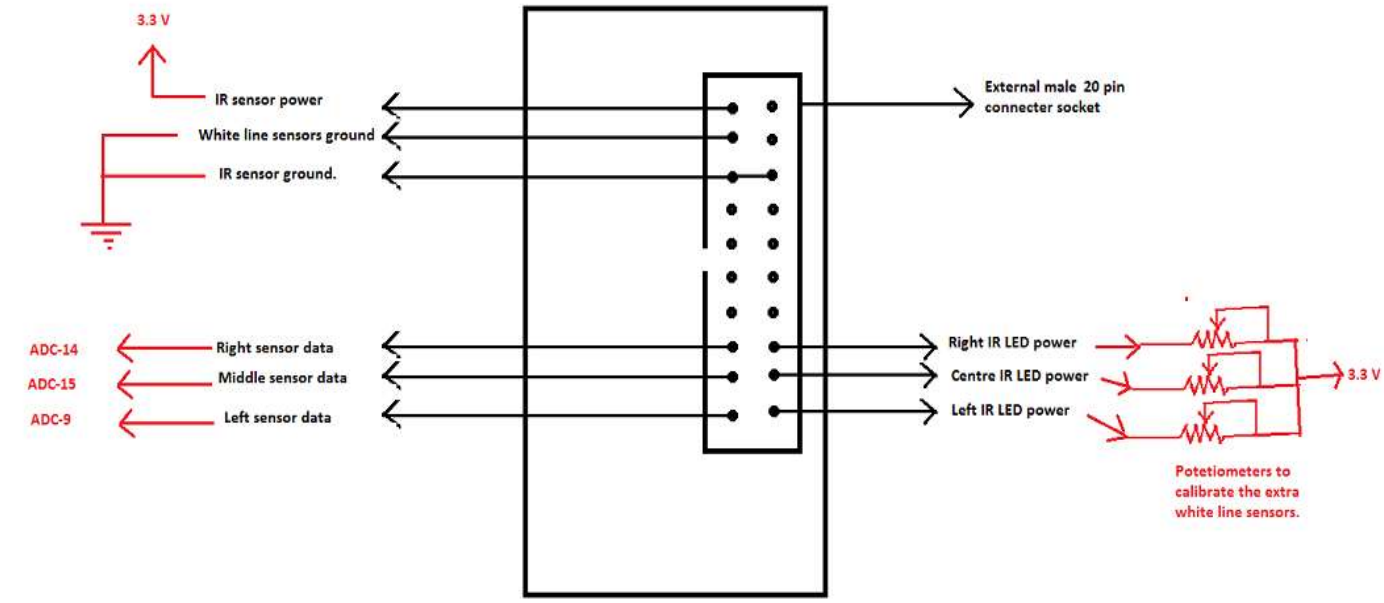


Fig 6



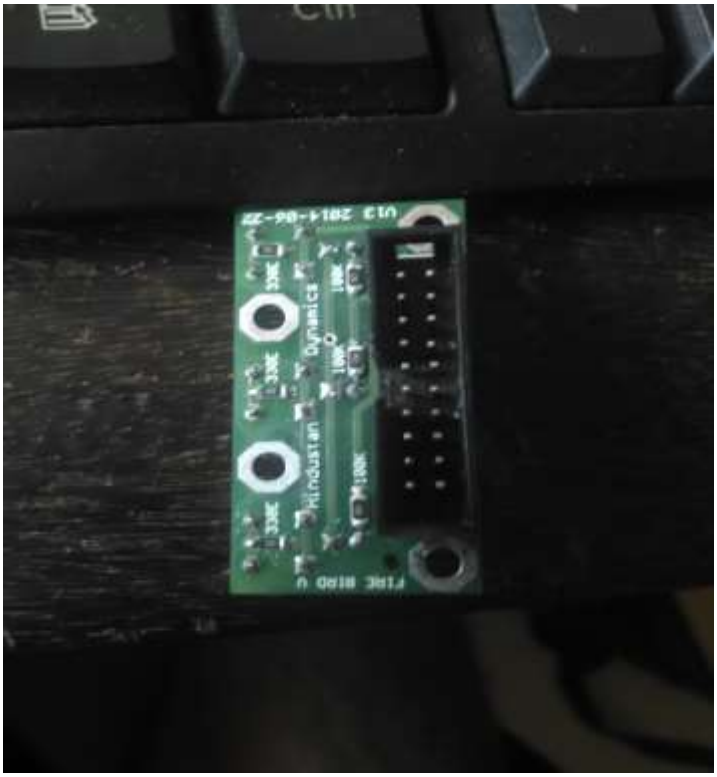


Fig 7

b. Upload a video to show the threshold values of white line sensor for detecting black and white colors of a block.

<

For video: Place the white line sensor near the block and display black and white values on the Liquid Crystal Display (LCD). Instructions to upload the video are given on the portal.
>

c. Threshold value obtained: white = ___50_____ Black = ___150_____

<
Determine the threshold value using the white line sensor values.
>

The you tube video can be found here
https://www.youtube.com/watch?v=mICzS0t6RLs&feature=youtu.be

The values given by the sensors for a white surface are around 20. So, we choose 50 to be the threshold value. Any value below 50 will be considered as a white surface.
The values given by the sensors for a black surface are around 200. So, we choose 150 to be the threshold value. Any value above 150 will be considered as a black surface.

Algorithm Analysis

Q-1 Draw a flowchart illustrating the algorithm used to complete the entire task. (50)

<
The flow chart should elaborate on every aspect of completing the entire task. Follow the standard pictorial representation used to draw the flowchart. There is no restriction on the number of pages for showing the complete algorithm
>

Since the flow charts are large, it is difficult to fit them into this document. So, the links to the flowchart are given below. (Kindly download the flowchart as viewing them in the drive gives less clarity. Also, refer to the explanation below for better understanding of the flowcharts.)

The algorithm follows the concept of "State machine" I, e, the next action to be performed depends on the outcomes/results of the current action.

The flowchart is divided into three parts-

1. Division D1

(https://drive.google.com/file/d/0BxjGlff4REI5X2Y2Ym5ITS1vREE/view?usp=sharing)

2. Section between D1 and D2.
(https://drive.google.com/file/d/0BxjGlff4REI5aXRFSVVtemE1aE0/view?usp=sharing)

3. Division D2.
(https://drive.google.com/file/d/0BxjGlff4REI5aTZTazdUNXZmNHc/view?usp=sharing)

4. The positions of the bot at the start and end of each section of the flowchart are as shown in the figure below.

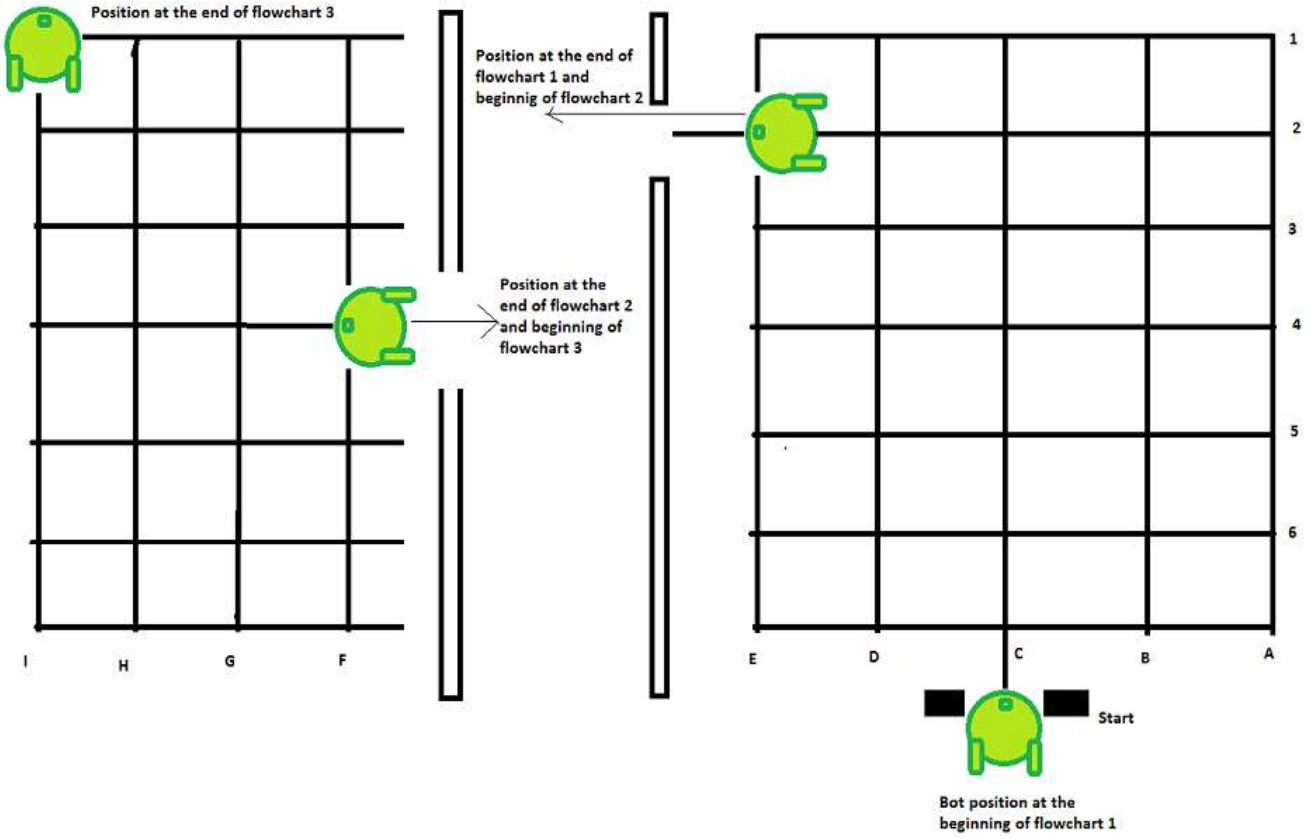


Fig 8

Briefing of the flowcharts:

1. For D1 : There are 4 variables used in the flowchart

1. next_action : This variable keeps track of the action that the bot has to perform when it encounters a node. Depending on the results of the action performed, the next action to be performed is determined and the value of this variable is appropriately updated.
2. row : This variable keeps track of the current row the bot is present at.
3. cube_flag_A : This variable is usually 0. But, in cases where there is one cube behind another cube, (for ex, a cube at (4, B) and (4, A)) the bot aligns the cube on column B (I, e, (4, A)) and detects the presence of the cube behind, on column A, and sets this variable if a cube is detected. Referring to this variable, the bot later reaches that position and aligns the cube.
4. cube_flag_E : This variable is usually 0. But, in cases where there is one cube behind another cube, (for ex, a cube at (4, D) and (4, E)) the bot aligns the cube on column D (I, e, (4, D) in our example) and detects the presence of the cube behind, on column E (I, e, (4, E) in our example), and sets this variable if a cube is detected. Referring to this variable, the bot later reaches that position and aligns the cube.

2. For the section between D1 and D2: Here, the algorithm is simple. As soon as the bot exits from D1, it starts following wall 1 with its gripper rotated towards right (facing wall 2) such that the extra sharp sensor mounted on the gripper is able to detect the gate into D2. The bot also keeps track of the distance for which travels while following wall 1 before it enters into D2. This is important in order to keep track of the row at which the gate would be opened. Depending on the distance travelled, the bot determines which row the gate is opened at.

3. For D2 : There are 2 variables used in the flowchart
    1. next_action : This variable keeps track of the action that the bot has to perform when it encounters a node. Depending on the results of the action performed, the next action to be performed is determined and the value of this variable is appropriately updated. Initially, the value of next_action is -1. This is the case when the bot has just entered D2. At the end of this action, the bot will be present at the 6th row on column F.
    2. row: This variable keeps track of the row at which the bot is present at any instant. The initial value of this variable depends on the gate that is open. The initial of row is calculated by the wall following section algorithm and passed on to this section of the algorithm.

A lot of abstraction is employed. For example, the “Follow line” call in the flowchart is itself a separate algorithm. To make the algorithm more understandable through the flowchart, functions like “Follow line”, “Follow wall”, “Align the cube” etc. have been directly used in the flowchart without giving any implementation details.