e-Yantra Robotics Competition - 2017

Theme and Implementation Analysis - Harvester Bot <eYRC#2361>

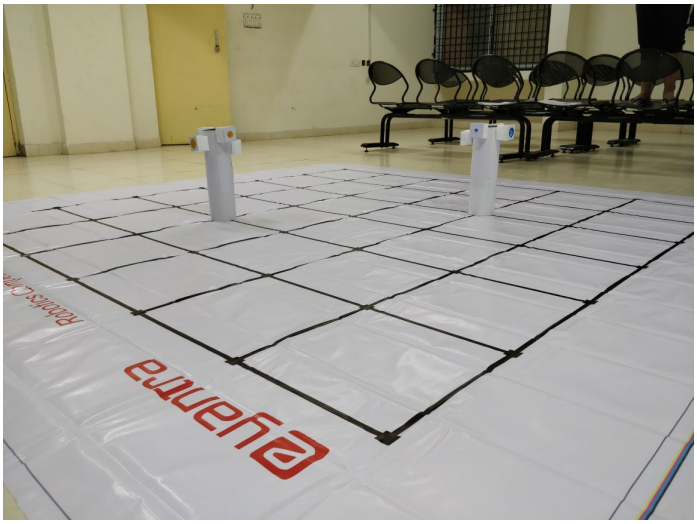
K PRANATH REDDY

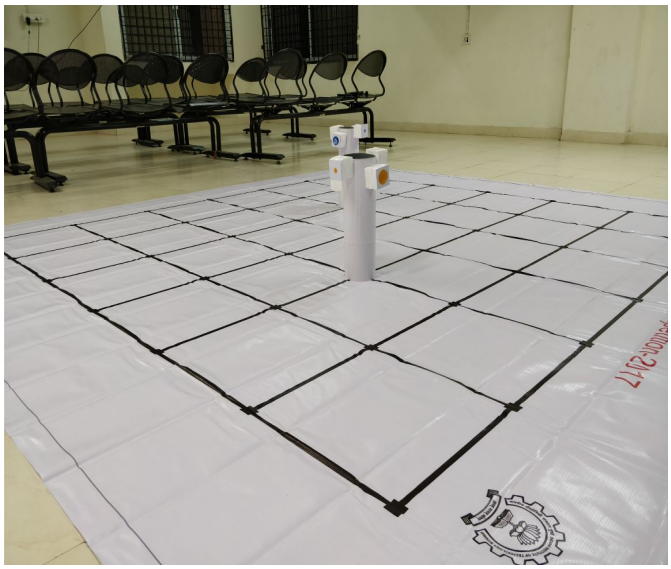
pranathreddy@outlook.com

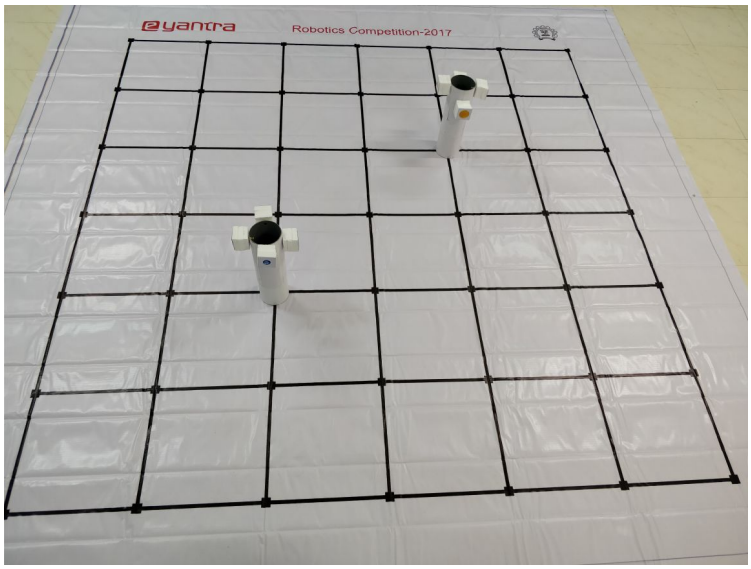
Scope and Preparing the Arena

Scope of the theme :

Harvester bots uses sensors to detect fruits, vegetables and other products. Further, these are helpful in picking up the good products ignoring all the immature ones. This saves a lot of time and strength for the farmers. Also, there are cases where people climb trees for collecting fruits and end up hurting themselves falling down from the tree. Also, some trees are home for many animals like snakes which might be harmful. Thus, the harvester bot recognises, collects and deposits the products as it is programmed. By using these bots, we can select methods to harvest fruits in a way such that the rough picking is avoided which resolves the problem of bruising and increases the quality and demand of the fruits. So, the purpose of this application is to save the time and strength utilised in harvesting







ARENA IMAGES

Building Modules

Major components required for designing the robotic system for the solution of the theme :

ELECTRONIC SYSTEMS:

1. Sharp IR sensors :- In designing the robotic system, we use sharp IR sensors to detect the obstacles (Trees and Deposition zones).

2.White line sensors :- We use white line sensors in the robotic system to sense the black lines, so that it can traverse the arena (black line following).

3.IR Proximity sensors :- we use an IR Proximity sensor to maintain a safe distance from the obstacle(tree/ fruit) to prevent collision with obstacle(tree).

4.Motor driver IC :- It is used to control motors in the our autonomous robotic system. Here we use L239D as motor driver IC.

4.Power supply unit :- It supplies electric power to robotic system.

5.Processing Unit :- It does mathematical computations using the data obtained from the sensors, also it takes decisions regarding the behaviour of the robotic system.

6.Signal conditioning unit :- It helps to convert one form of a signal to another. For example, ADC converts analog signals (received from sensors) to digital signal.

MECHANICAL SYSTEMS:

1.DC Geared Motor :- These are attached to the wheels of the robot which helps for the primary motion of the robot along the arena.

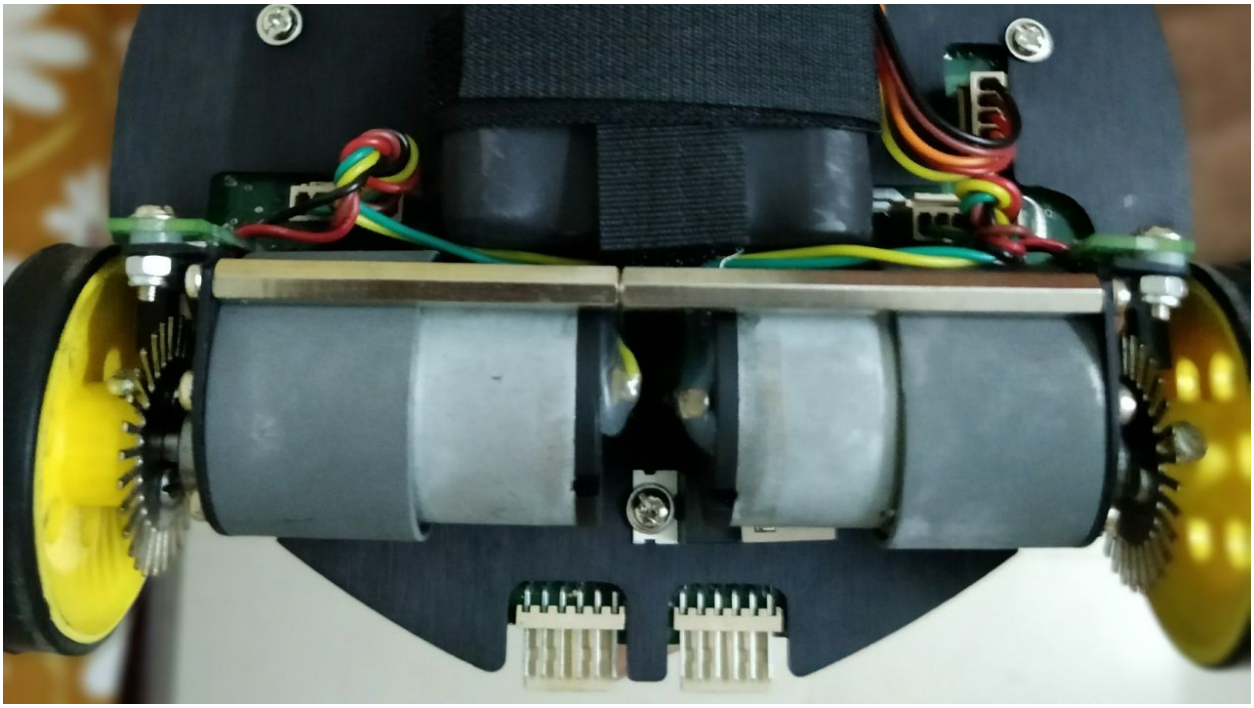
2.Servo Motors:- These motors are used in the plucking mechanism of robotic system to move the robotic arm front and back.

Actuators

The actuators present on Firebird V robot and additional actuators required for designing the bot

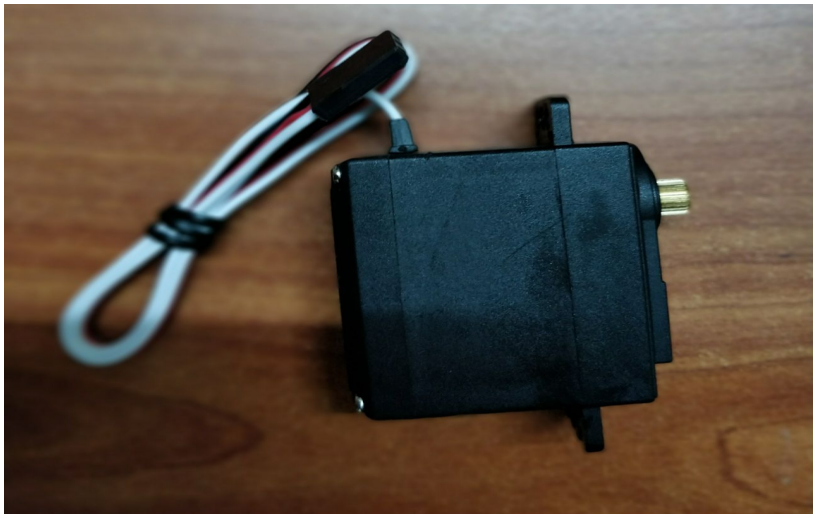
Actuators Present on Firebird V :

1. DC geared Motors :- Firebird V has two DC geared Motors in differential drive configuration and castor wheel at front as support.



Additional Actuators :

2.Servo Motors :- A servo motor is a rotary actuator that allows for precise control of angular position



Power Management

The power management system required for a robot in general and for the autonomous robot used for this theme :

Firebird V is powered in two ways,

(i) battery power

(ii)Auxiliary power

We use 'battery power mode' to power up firebird V. We will use this mode because our robotic system requires portability.Fire Bird V is powered by 9.6V, 2.1Ah rechargeable Nickel Metal Hydride battery pack.

Actuators :-

DC geared Motors :- DC geared Motors are powered by 'V mot supply', which is varied between 8V to 11.7V depending on the battery state.5V system supply is used for driving L293D.Motors are controlled by L293D dual motor driver to which can provide upto 600mA of current to each motor.

Servo Motors :-

StandBy current consumption : 100mA

operational current consumption : 1-2A

Sensors:-

Power to the sensors which are on-board the fire bird V is supplied by the battery connected '5V system' supply.

Sharp IR sensor :-

Max Ratings

the supply voltage range is -0.3V to 7V

output terminal voltage: -0.3V to 0.3V

operational values:

Supply voltage: 4.5V to 5.5V

output terminal voltage: 0.25V to 0.55V

average current supply : 33mA to 50mA

LCD Display:

Low power operational voltage : 2.7V to 5.5V

wide range LCD Driver voltage : 3V to 11V

L293D:

Maximum ratings :

collector supply volatage: 36V

Logic supply volatage : 4.5 to 36V

input voltage : 7V

Power dissipation at 80C = 5W

Design Analysis

Position of the mechanism for identifying, plucking and depositing the Fruits in the Deposition Zones :

1. If arm is placed at the front, it obstructs the line of sight of the sharp IR sensor, thus hindering its functionality. Also, if it is placed at front, structurally it becomes less stable and tends to topple.

2. If the arm is placed on either of the sides, it obstructs the line of sight of proximity sensors, thus hindering their functionality.

   Thus, the best option is to place the arm on the back.

The design of the mechanism and how it is mounted on the robot :

The mechanism for plucking and collecting the fruits of a tree shall consist of a rear mounted two arm mechanism with the camera mounted on the upper arm which shall rise up slowly. The images captured by the camera are sent for processing and when the image matches that of the fruit to be plucked, the arm shall move forward towards the fruit. The mechanism also consists of a plucker which helps in plucking the fruit. Once the fruit is plucked on one side of the tree, it shall be dropped in the basket at the rear of bot and move towards next side of the tree where fruits are present(at 90 degrees to the initial position) and repeats the process for identifying and plucking the fruits. This process is repeated until the bot collects fruits from all accessible sides and puts it in the basket.

Challenges expected in designing the mechanism for identifying, plucking and dropping the Fruits :

Challenge: Calibrating the arm to exactly reach the fruit, without hitting other parts of the tree.

Solution: the arm consists of the camera which shall rise to a certain height initially without hitting other parts of the tree

Challenge: Identifying the required fruit.

Solution: we use image processing methods to identify the fruit

Challenge: Placing the plucked fruit in the basket.

Solution: the arm rotates 180 degrees after plucking the fruit and leaves the fruit from the plucker so that it falls into the basket

Challenge : Plucking the fruit from the tree.

Solution : the plucker fruit holds onto the fruit and then the bot moves backward slowly so that the fruit is deattached from the tree

the actuators used in designing the mechanism :

Servo motor provides precise movements thus making it possible to capture image and check for the presence of a fruit at different heights of the tree.

DC motor is less precise than servo motor / not precise enough for the required operation mentioned above.

Stepper motor provides a highly accurate motion as it has very small angle of rotation. This much accuracy is not essential and hence leads slow detection of fruits thus increasing the time required to complete job.

Environment Sensing

Use of the provided sensors to implement the theme :

Sensor 1: Sharp IR sensor

It is used to detect obstacle and calculate it's distance from the bot.

Sensor 2: Proximity sensor

It is used to find the presence of fruit on different sides of the tree and finding their height from the top. It also helps to avert the arm hitting other objects.

Sensor 3: White line sensor

It is used to make the bot follow the black line. Black and white colours reflect different magnitudes of incident light. Based on this principle, the difference in amount of light reflected back to the sensor, it can be concluded whether the bot is moving on black line or not.

sensor 4: camera

it is used to detect the fruits by using image processing methods

Communication

Method/s of communication between:

A) Firebird V robot and Raspberry-Pi

B) Computer and Raspberry-Pi

A) **Firebird V robot and Raspberry-Pi** :

Firebird V can communicate with other robots/ devices(eg. Raspberry pi) serially using either wired link/ wireless module. Serial communication is done in asynchronous mode.We use Serial Communication on UART1 for the communication between Firebird V robot and Raspberry-Pi. We connect serial pins of Firebird V to the GPIO pins of raspberry-Pi for the serial communication.

B) **Computer and Raspberry-Pi** :

Raspberry Pi can be connected to the computer using an ethernet cable or via WiFi. The communication between the computer and the Raspberry-Pi is done using the (Secure shell)SSH protocol which is a method for secure remote login from one computer to another.

Algorithm Analysis

Brief Algorithm :

Components required for navigation

i. Three white line sensors
ii. Sharp IR sensor
iii. position encoder

The navigation of the bot is primarily down by using the white line sensors present on the bottom of the bot and we also use the sharp IR sensor present in the front side of the bot which helps in detecting obstacles and trees. We will give the initial direction, tree positions, deposition nodes as the inputs to the bot. We first Sort the distances of the trees, then select the adjacent nodes of each tree individually and again sort them, now take the first element of each sorted list of the adjacent nodes of the trees and arrange them in the order corresponding to the sorted list of distances of trees that was found initially,This will give us the target nodes in the order of their distances. If we imagine each node of the arena to be a point in X-Y plane, Once the target node is selected we find the difference of the x-coordinates and y-coordinates the present node and the target node if the initial direction is north we first make difference of y-coordinates zero and then turn the robot into the x-axis and make the difference of the x-coordinates to be zero, thus the bot will reach the target node. Similarly we follow the same steps with three other cases of initial direction accordingly. In case the bot detects an obstacle in its path with the help of Sharp IR sensor rotates 90 degrees into the axis perpendicular to the one it is travelling in and sets an intermediate target node and it first reaches this intermediate target node which we set it to be adjacent to the obstacle and towards the target node thus avoiding collision with the obstacle the bot continues to reach the target node. Thus the bot reaches all the target nodes and deposition nodes and finishes the required task. This is the navigation scheme proposed for this theme.

***** A Detailed algorithm is attached along with the document*****

Challenges

Major challenges anticipated in addressing this theme :

Challenge-1 : Tree/Deposition Zone detection and reaching the trees in shortest way.

Solution : Given the (x,y) positions of the trees and deposition zones, we develop an algorithm to sort the distances of each tree from the current position to identify which tree is nearest from the current position. After identifying the nearest tree, a target node is identified and the bot travels towards the target tree, detecting the obstacles in it's path and averting them.

Challenge-2 : After detecting the tree, robot has to decide whether it is a collected tree / uncollected tree / deposition zone.

Solution : Our algorithm consists of a mechanism to identify the tree nearest to the current position. So the bot will consider only that tree for plucking and all other things (trees and deposition zones) as obstacles.

Challenge-3 : One of the major challenge is to design a mechanism for plucking the fruit and collecting all fruits of one tree at a time.

Solution : The arm consists of a clipper mechanism which holds the fruit tightly. Once the fruit is held properly, the bot moves back, thus pulling the fruit out of the tree. Once the fruit is plucked, the arm rotates towards the basket which holds all the fruits collected from the tree and drops into it.

Challenge-4 : Depositing the fruit in the deposition zone such that no fruit falls outside

Solution : We design a basket which stores all the plucked fruits and after reaching the deposition zone this basket is dumped so that the fruits do not fall out of the on zone.

Challenge-5 : Preventing drainage of excess power by considering voltage and current ratings of the sensors and motors.

Solution : By switching the not in use components from operation mode to standby mode, we reduce current consumption by 300mA.

For example, we supply power to the arm only when the bot detects the tree.

**********THIS WAY IT WILL TRAVESE THROUGH ARENA**********

NAVIGATION ALGORITHM - eYRC#2361

1. Start at node 0 facing E-YANTRA logo.
2. Initial_Direction, tree positions, Deposition zones are given as inputs.
3. Read present node as (Xp,Yp)=(0,0).
4. Distance is calculated using distance formula
   4.1. Sort the distances of the trees and then select the adjacent nodes of each tree individually and again sort them, now take the first element of each sorted list of the adjacent nodes of the trees and arrange them in the order corresponding to the sorted list of distances of trees that was found initially,This will give us the target nodes in the order of their distances .Now using this newly found list set the target node as (Xd,Yd)
5. While Right_whiteline sensor >= 100 and Central_whiteline sensor>=100 and Left_whiteline sensor>=100
   5.1 (Nx,Ny)=(Xd-Xp,Yd-Yp)
   5.2 While(Nx!=0 or Ny!=0)
      5.2.1 Move 3cm forward in the direction of initial direction.
      5.2.2 If Nx=0
         5.2.2.1 If initial direction=east
            5.2.2.1.1 If Ny > 0
               5.2.2.1.1.1 Turn 90 degrees anticlockwise
               5.2.2.1.1.2 Initial direction = north
               5.2.2.1.1.3 ON sharp IR sensor and read the value as k.
               5.2.2.1.1.4 If k >40
                  5.2.2.1.1.4.1 move to next node in initial direction
                  5.2.2.1.1.4.2 Ny = Ny-1
               5.2.2.1.1.5 If k<40
                  5.2.2.1.1.5.1 turn 90 degrees clock wise
                  5.2.2.1.1.5.2 initial direction = east
                  5.2.2.1.1.5.3 move to next node in initial direction
                  5.2.2.1.1.5.4 turn 90 degrees anticlockwise
                  5.2.2.1.1.5.5 initial direction = north
                  5.2.2.1.1.5.6 move two nodes in initial direction
                  5.2.2.1.1.5.7 turn 90 degrees anticlockwise
                  5.2.2.1.1.5.8 initial direction = west
                  5.2.2.1.1.5.9 move forward to next node
                  5.2.2.1.1.5.10 turn 90 degrees clockwise
                  5.2.2.1.1.5.11 initial direction = north
                  5.2.2.1.1.5.12 Go-to 5.2.2.1.1.3
            5.2.2.1.2 if Ny < 0
               5.2.2.1.2.1 Turn 90 degrees clockwise
               5.2.2.1.2.2 Initial direction = south
               5.2.2.1.2.3 ON sharp IR sensor and read the value as k.
               5.2.2.1.2.4 If k>40
                  5.2.2.1.2.4.1 move to next node in initial direction
                  5.2.2.1.2.4.2 Ny = Ny+1

5.2.2.1.2.5 if k<40

5.2.2.1.2.5.1 turn 90 degrees clock wise

5.2.2.1.2.5.2 initial direction = west

5.2.2.1.2.5.3 move to next node in initial direction

5.2.2.1.2.5.4 turn 90 degrees anticlockwise

5.2.2.1.2.5.5 initial direction = south

5.2.2.1.2.5.6 move two nodes in initial direction

5.2.2.1.2.5.7 turn 90 degrees anticlockwise

5.2.2.1.2.5.8 initial direction = east

5.2.2.1.2.5.9 move forward to next node

5.2.2.1.2.5.10 turn 90 degrees clockwise

5.2.2.1.2.5.11 initial direction = south

5.2.2.1.2.5.12 Go-to 5.2.2.1.2.3

5.2.2.2 if initial direction=west

5.2.2.2.1 If Ny > 0

5.2.2.2.1.1 Turn 90 degrees clockwise

5.2.2.2.1.2 Initial direction = north

5.2.2.2.1.3 ON sharp IR sensor and read the value as k.

5.2.2.2.1.4 If k >40

5.2.2.2.1.4.1 move to next node in initial direction

5.2.2.2.1.4.2 Ny = Ny-1

5.2.2.2.1.5 If k<40

5.2.2.2.1.5.1 turn 90 degrees clock wise

5.2.2.2.1.5.2 initial direction = east

5.2.2.2.1.5.3 move to next node in initial direction

5.2.2.2.1.5.4 turn 90 degrees anticlockwise

5.2.2.2.1.5.5 initial direction = north

5.2.2.2.1.5.6 move two nodes in initial direction

5.2.2.2.1.5.7 turn 90 degrees anticlockwise

5.2.2.2.1.5.8 initial direction = west

5.2.2.2.1.5.9 move forward to next node

5.2.2.2.1.5.10 turn 90 degrees clockwise

5.2.2.2.1.5.11 initial direction = north

5.2.2.2.1.5.12 Go-to 5.2.2.2.1.3

5.2.2.2.2 if Ny < 0

5.2.2.2.2.1 Turn 90 degrees anticlockwise

5.2.2.2.2.2 Initial direction = south

5.2.2.2.2.3 ON sharp IR sensor and read the value as k.

5.2.2.2.2.4 If k>40

5.2.2.2.2.4.1 move to next node in initial direction

5.2.2.2.2.4.2 Ny = Ny+1

5.2.2.2.2.5 if k<40

5.2.2.2.2.5.1 turn 90 degrees clock wise

5.2.2.2.2.5.2 initial direction = west

5.2.2.2.2.5.3 move to next node in initial direction
5.2.2.2.2.5.4 turn 90 degrees anticlockwise
5.2.2.2.2.5.5 initial direction = south
5.2.2.2.2.5.6 move two nodes in initial direction
5.2.2.2.2.5.7 turn 90 degrees anticlockwise
5.2.2.2.2.5.8 initial direction = east
5.2.2.2.2.5.9 move forward to next node
5.2.2.2.2.5.10 turn 90 degrees clockwise
5.2.2.2.2.5.11 initial direction = south
5.2.2.2.2.5.12 Go-to 5.2.2.2.2.3

5.2.3 if Ny=0

5.2.3.1 If initial direction=north

5.2.3.1.1 If Nx > 0

5.2.3.1.1.1 Turn 90 degrees clockwise
5.2.3.1.1.2 Initial direction = east
5.2.3.1.1.3 ON sharp IR sensor and read the value as k.
5.2.3.1.1.4 If k >40
5.2.3.1.1.4.1 move to next node in initial direction
5.2.3.1.1.4.2 Nx= Nx-1
5.2.3.1.1.5 If k<40
5.2.3.1.1.5.1 turn 90 degrees clock wise
5.2.3.1.1.5.2 initial direction = south
5.2.3.1.1.5.3 move to next node in initial direction
5.2.3.1.1.5.4 turn 90 degrees anticlockwise
5.2.3.1.1.5.5 initial direction = east
5.2.3.1.1.5.6 move two nodes in initial direction
5.2.3.1.1.5.7 turn 90 degrees anticlockwise
5.2.3.1.1.5.8 initial direction = north
5.2.3.1.1.5.9 move forward to next node
5.2.3.1.1.5.10 turn 90 degrees clockwise
5.2.3.1.1.5.11 initial direction = east
5.2.3.1.1.5.12 Go-to 5.2.3.1.1.3

5.2.3.1.2 if Nx < 0

5.2.3.1.2.1 Turn 90 degrees anticlockwise
5.2.3.1.2.2 Initial direction = west
5.2.3.1.2.3 ON sharp IR sensor and read the value as k.
5.2.3.1.2.4 If k>40
5.2.3.1.2.4.1 move to next node in initial direction
5.2.3.1.2.4.2 Nx = Nx+1
5.2.3.1.2.5 if k<40
5.2.3.1.2.5.1 turn 90 degrees clock wise
5.2.3.1.2.5.2 initial direction = north
5.2.3.1.2.5.3 move to next node in initial direction
5.2.3.1.2.5.4 turn 90 degrees anticlockwise
5.2.3.1.2.5.5 initial direction = west
5.2.3.1.2.5.6 move two nodes in initial direction

5.2.3.1.2.5.7 turn 90 degrees anticlockwise
5.2.3.1.2.5.8 initial direction = south
5.2.3.1.2.5.9 move forward to next node
5.2.3.1.2.5.10 turn 90 degrees clockwise
5.2.3.1.2.5.11 initial direction = west
5.2.3.1.2.5.12 Go-to 5.2.3.1.2.3

5.2.3.2 If initial direction=south

5.2.3.2.1 If Nx > 0

5.2.3.2.1.1 Turn 90 degrees anticlockwise
5.2.3.2.1.2 Initial direction = east
5.2.3.2.1.3 ON sharp IR sensor and read the value as k.
5.2.3.2.1.4 If k >40
5.2.3.2.1.4.1 move to next node in initial direction
5.2.3.2.1.4.2 Ny = Ny-1
5.2.3.2.1.5 If k<40
5.2.3.2.1.5.1 turn 90 degrees clock wise
5.2.3.2.1.5.2 initial direction = south
5.2.3.2.1.5.3 move to next node in initial direction
5.2.3.2.1.5.4 turn 90 degrees anticlockwise
5.2.3.2.1.5.5 initial direction = east
5.2.3.2.1.5.6 move two nodes in initial direction
5.2.3.2.1.5.7 turn 90 degrees anticlockwise
5.2.3.2.1.5.8 initial direction = north
5.2.3.2.1.5.9 move forward to next node
5.2.3.2.1.5.10 turn 90 degrees clockwise
5.2.3.2.1.5.11 initial direction = east
5.2.3.2.1.5.12 Go-to 5.2.2.1.1.3

5.2.3.2.2 if Ny < 0

5.2.3.2.2.1 Turn 90 degrees clockwise
5.2.3.2.2.2 Initial direction = south
5.2.3.2.2.3 ON sharp IR sensor and read the value as k.
5.2.3.2.2.4 If k>40
5.2.3.2.2.4.1 move to next node in initial direction
5.2.3.2.2.4.2 Ny = Ny+1
5.2.3.2.2.5 if k<40
5.2.3.2.2.5.1 turn 90 degrees clock wise
5.2.3.2.2.5.2 initial direction = west
5.2.3.2.2.5.3 move to next node in initial direction
5.2.3.2.2.5.4 turn 90 degrees anticlockwise
5.2.3.2.2.5.5 initial direction = south
5.2.3.2.2.5.6 move two nodes in initial direction
5.2.3.2.2.5.7 turn 90 degrees anticlockwise
5.2.3.2.2.5.8 initial direction = east
5.2.3.2.2.5.9 move forward to next node
5.2.3.2.2.5.10 turn 90 degrees clockwise
5.2.3.2.2.5.11 initial direction = south

5.2.3.2.2.5.12 Go-to 5.2.3.2.2.3

5.2.4 ON sharp IR sensor and read the value as k.

5.2.5 If k>40

5.2.5.1 If initial_direction =North

5.2.5.1.1.1 If Ny>=0

5.2.5.1.1.1.1 Move to next node in initial direction

5.2.5.1.1.1.2 Ny=Ny-1.

5.2.5.1.1.2 If Ny<0

5.2.5.1.1.2.1 Rotate 180 degrees clockwise

5.2.5.1.1.2.2 Initial direction = south

5.2.5.2 If initial_direction =South

5.2.5.2.1 If Ny>=0

5.2.5.2.1.1 Rotate 180 degrees clockwise

5.2.5.2.1.2 Initial direction = north

5.2.5.2.2 Ny<0

5.2.5.2.2.1 Move to next node in initial direction

5.2.5.2.2.2 Ny=Ny+1.

5.2.5.3 If initial direction = east

5.2.5.3.1 If Nx>=0

5.2.5.3.1.1 Move to next node in initial direction

5.2.5.3.1.2 Nx = Nx-1.

5.2.5.3.2 If Nx < 0

5.2.5.3.2.1 Rotate 180 degrees clockwise

5.2.5.3.2.2 Initial direction = west

5.2.5.4 If initial direction = west

5.2.5.4.1 If Nx>=0

5.2.5.4.1.1 Rotate 180 degrees clockwise

5.2.5.4.1.2 Initial direction = east

5.2.5.4.2 If Nx<0

5.2.5.4.2.1 Move to next node in initial direction

5.2.5.4.2.2 Nx=Nx+1

5.2.6 If k<40

5.2.6.1 If initial direction = North

5.2.6.1.1 Turns 90 degrees clockwise

5.2.6.1.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.1.2.1 Initial direction = east

5.2.6.1.2.2 Go-to step 5.2.2

5.2.6.1.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.1.3.1 Turn 90 degrees clockwise

5.2.6.1.3.2 if Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.1.3.2.1 initial direction = south

5.2.6.1.3.2.2 Go-to step 5.2.2

5.2.6.1.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.1.3.3.1 Turn 90 degrees clockwise

5.2.6.1.3.3.2 if Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.1.3.3.2.1 initial direction = west

5.2.6.1.3.3.2.2 Go-to step 5.2.2

5.2.6.1.3.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.1.3.3.3.1 stop

5.2.6.2 If initial direction = south

5.2.6.2.1 Turn 90 degrees clockwise

5.2.6.2.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.2.2.1 initial direction = west

5.2.6.2.2.2 Go-to step 5.2.2

5.2.6.2.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.2.3.1 Turn 90 degrees clockwise

5.2.6.2.3.2 if Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.2.3.2.1 initial direction = north

5.2.6.2.3.2.2 Go-to step 5.2.2

5.2.6.2.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.2.3.3.1 Turn 90 degrees clockwise

5.2.6.2.3.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.2.3.3.2.1 initial direction = east

5.2.6.2.3.3.2.2 Go-to step 5.2.2

5.2.6.2.3.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.2.3.3.3.1 stop

5.2.6.3 If initial direction = east

5.2.6.3.1 Turn 90 degrees clockwise

5.2.6.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.3.2.1 initial direction = south

5.2.6.3.2.2 Go-to step 5.2.2

5.2.6.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.3.3.1 Turn 90 degrees clockwise

5.2.6.3.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.3.3.2.1 initial direction = west

5.2.6.3.3.2.2 Go-to step 5.2.2

5.2.6.3.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.3.3.3.1 Turn 90 degrees clockwise

5.2.6.3.3.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.3.3.3.2.1 initial direction = north

5.2.6.3.3.3.2.2 Go-to 5.2.2

5.2.6.3.3.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.3.3.3.3.1 Stop

5.2.6.4 I initial direction = west

5.2.6.4.1 Turn 90 degrees clockwise

5.2.6.4.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.4.2.1 initial direction = north

5.2.6.4.2.2 Go-to step 5.2.2

5.2.6.4.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.4.3.1 Turn 90 degrees clockwise

5.2.6.4.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor>=100, Left_whiteline sensor<100

5.2.6.4.3.2.1 initial direction = east

5.2.6.4.3.2.2 Go-to step 5.2.2

5.2.6.4.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.4.3.3.1 Turn 90 degrees clockwise

5.2.6.4.3.3.2 If Right_whiteline sensor < 100, Central_whiteline sensor >=100, Left_whiteline sensor<100

5.2.6.4.3.3.2.1 initial direction = south

5.2.6.4.3.3.2.2 Go-to step 5.2.2

5.2.6.4.3.3.3 If Right_whiteline sensor < 100, Central_whiteline sensor <=100, Left_whiteline sensor<100

5.2.6.4.3.3.3.1 Stop

6. Update the present node (Xp,Yp) = (Xd,Yd)
7. Update the target node (Xd,Yd) by finding the next tree using the sorted distances in step 4
8. Go to step 6 and continue.
9. The program runs until the bot finishes the task.

**********THIS WAY IT WILL TRAVESE THROUGH ARENA**********