

**e-Yantra Robotics Competition - 2018**

**Theme and Implementation Analysis – Thirsty Crow**

**1079**

|  |  |
| --- | --- |
| **Team leader name** | **Sourabh Khemka** |
| **College** | **Birla Institute of Technology, Mesra** |
| **Email** | **sourabhkhemkask@gmail.com** |
| **Date** | **30-01-2019** |

**Scope**

**Q1 a. State the scope of the theme assigned to you. (5)**

The theme Thirsty Crow is based on story of crow and water pitcher.

The theme provides knowledge on various aspects like AR, image processing, microcontroller with the help of a tale which says thoughtfulness is superior to strength and also shows application of robotics on nature.

Implementation includes traversing a crow-bot in an arena and perform task of picking-up/depositing pebbles from pick-up/deposition zones in arena. The theme has to be realized with help of augmented reality. The bot and pickup/deposition zones have aruco markers on which models have to be shown in the computer screen along with required animations.

**Testing your knowledge (theme and rulebook analysis)**

**Q2. a) What is considered a correct pebble pick-up? (2.5)**

**b) What is considered a correct pebble drop? (2.5)**

The crow-robot has to pick-up pebble from pickup locations i-e below the pebble aruco marker (markers with ID ranging from 1-9). The bot has to reach a pebble location and use the electromagnetic pickup mechanism to pick up one pebble at a time and when the bot picks up a pebble the pick-up has to be depicted using augmented reality by diminishing of the pebble’s 3-D model above the aruco marker at that pick-up location. When this is successfully achieved, it is considered a correct pebble pick-up.

The crow-bot after picking up the pebble from the pick-up location, has to traverse to the deposition zone i-e below the pitcher aruco marker (marker with ID 0). After reaching the location of the pitcher, bot has to drop the pebble within the deposition zone under the pitcher aruco marker by switching off the power supply of the electromagnet. The dropping of pebble has to be shown by increase in the water level in the pitcher after the pebble has been dropped. When this is successfully achieved, it is considered as a correct pebble drop.

**Mechanism**

**Q3. Explain the mechanism that you would use to pick and drop the pebbles. (10)**

The bot uses a long forward pointing rectangular arm. The electromagnet is clamped at the end of the arm with a screw penetrating the arm into the threads provided on the top circular face of the electromagnet. The electromagnet is placed such that the attracting surface is at an angle with the vertical to the ground. This increases the reach of the electromagnet (makes the approach better and less prone to failure due to miscalculations) and the ground clearance for the pebble.

Bolt

Electromagnet

Rectangular arm rod

Pebble

Figure representing the electromagnet’s mounting

The arm length is adjusted such that it does not collide with the aruco marker while traversing the arena after pick-up/deposit of the pebble. The rectangular arm is thick enough to provide it strength of holding up the electromagnet at the required distance. The arm length cannot be very long as this makes it prone to collide with the aruco marker while turning.

The entire process would be as follows: the crow-bot arrives at the proper node in front of the pickup zone. The bot would advance further such that the arm with the electromagnet attached to it comes within pickup zone. Now, the electromagnet is over the pebbles and it is turned on. The magnet’s strength is sufficiently enough to attract one pebble at a time (as required for theme implementation). Thus, it picks up a pebble and the bot comes backward to the node and traverses towards the deposit zone. The magnet is ON throughout the traversal for holding up the pebble. The same process is repeated at the deposit zone and when the electromagnet is within the deposit zone, it is turned off to drop the pebble. The bot comes back to the node and continues the further traversal.

**Algorithm Analysis**

**Q4. Draw a flowchart illustrating the algorithm you propose to use for theme implementation. (10)**

**START**

Python script is run which uses Open-GL functions to render 3-D models of crow, pebbles and pitcher on their respective aruco markers

**The robot now has to move in any one of the available paths from the current node (which can also be the start position)**

**Traversal algorithm will start with the pick-up position at the 0th index of the list as target location**

J

A

Python function is called to detect the aruco markers at pick-up and deposition zones. The function then stores IDs of pick-up locations in a list **pickup\_locations** and deposition zone in a variable

Python function is called to calculate Euclidean distance between bot and all pick-up locations and to **append distance values (if function is being called first time)/ refresh the values (if being recalled)** to the list pickup\_locations corresponding to value of IDs (added in the second step of the flow-chart)

Function to **sort** the list **pickup\_locations** with **Euclidean** **distance** as **key** is called

K

A

K

J

***Traversal function*** **is called** which ***takes***target location i-e ***the aruco ID at top of list pickup\_locations as input*** and ***returns the path which bot has to take from current node***

Character is sent to the microcontroller using serial communication which signifies the path to be taken

Microcontroller receives the character using the UART\_rx() function

Microcontroller calls function to turn bot in required direction

Motors are run until a certain encoder counts are reached and then the white line sensor is used to align the bot with the path

forward () function is called in the microcontroller

Stop() function is called by microcontroller when bot reaches next node

If

Node reached is of hexagon of a pick-up zone

False

H

B

True

H

B

Python function to find the node of the hexagon around pick-up zone on which bot needs to be placed will be called. The function returns a string to direct the microcontroller to how much to move the bot (either clockwise or anti-clockwise) along the hexagon to reach the required node.

Python sends the string to microcontroller using X-Bee modules

Microcontroller receives the string using the UART\_rx() function

Microcontroller follows the instructions received and takes the bot to required node of that hexagon

The **function to** move the bot to **pick-up** the pebbles is called

A Python function is called which replaces the 3-D model of pebbles with a model of diminished pebbles

**Traversal function is called** in Python with list containing **ID of deposition zone stores in a variable as input**

The function returns instructions to take the bot to **next node** which is sent to microcontroller using UART

G

C

F

G

C

F

If

The node reached (which was represented as **next node** in previous step) correspond to deposition zone

False

True

Python function to find the node of the hexagon around deposition zone on which bot needs to be placed will be called. The function returns a string to direct the microcontroller to how much to move the bot (either clockwise or anti-clockwise) along the hexagon to reach the required node.

Microcontroller receives the string using the UART\_rx() function

Microcontroller takes the bot to required node

Function to deposit the pebble is called in the AVR program

Python function to refresh the 3-D model of pitcher with a model of pitcher with increased water level

E

D

E

Python function is called which removes the ID of pick-up zone completed just now from the list pickup\_locations

D

If

List pickup\_locations is empty list

False

True

A character is sent from X-bee module connected to laptop to X-Bee module connected to micro-controller

On receiving the previously specified character microcontroller calls the function to sound the buzzer for 5 seconds

**STOP**

**Q5. What kind of path planning algorithm will you use in order to navigate your robot inside the arena? (10)**

Initially the bot is placed at start position and the markers for pitcher and pebbles are also placed at proper positions.

With the help of aruco markers above the crow-bot, pickup and deposition position we get the coordinates of the bot and the pickup/deposit position (we can get the position of the markers in the window and with help of aruco-IDs the coordinates of the markers can be classified as that of the crow-bot or pickup position or deposit position).

Let’s imagine a situation, the crow-bot is placed at the start-2 position and the pick-up position where the bot has to reach is placed in the center position. For the purpose of explanation let’s imagine that initial coordinate of crow-bot (when initially placed at start position) is **(20, 10)** and the coordinate of the pickup position at center hexagon is **(10, 10)**.

The crow-bot has to reach the pick-up position traversing the shortest path.

The coordinate of bot is **(20, 10)** and desired coordinate is the nearest possible to **(10, 10)** hence the bot has to move so as to reduce its abscissa. In this situation when bot is at start position and pickup position is at center of the arena, the bot has two possible available paths from the start position and both are identical so the bot moves in any one of the two\*.

The bot has now reached a node and has four options of path available to take. The current x-coordinate of bot is something less than 20 and greater than 10 and the y-coordinate is also unequal to 10 say >10.

At this node the bot has to move so as to reduce its x-coordinate as well as the y-coordinate. Python script sends a character to the microcontroller to actuate the motors accordingly.

The above algo keeps on executing in a loop until we reach a situation of minimum possible difference of coordinates of the crow-bot and the pick-up position.

The bot has now reached the desired hexagon and now it has to position itself at the correct node on that hexagon in order to be able to pick-up or drop the pebbles. This can be achieved with help of the rotation vector for the aruco marker of the crow-bot and the pickup/drop position. The python script will send commands to the microcontroller through serial communication using Xbee modules to position the bot on proper position on the hexagon so as to be able to pick-up/drop the pebbles using electromagnet pickup mechanism.

For the situation when the bot is at start position and we have multiple pick-up positions, the bot is required to reach the nearest pick-up position first. The Python script will be written to determine the id of nearest aruco marker by calculating and comparing the Euclidean distance between the coordinates of bot and all markers. The algo for traversal will the run for that id only. After the pebble is picked up the Python script will direct the traversal between the current position and the pitcher. After the pebble has been dropped the Python script will now direct the traversal from the deposit position to the nearest pick-up position from there.

The above algorithm has been chosen as the algorithm always moves the bot in a way to minimize the difference between the current position and the desired position, this prevents the bot from opting paths at the junction which increases the distance between the current and desired position. The above algorithm is also preferred as it does not demand for any extra sensor to detect the pick-up/drop positions and the use of aruco markers to detect the position is also based upon the previous knowledge and codes and doesn’t require image processing tasks such as contour detection, thresholding etc. which will increase the complexity and also the chances of error.

**Challenges**

**Q6. What are the major challenges that you can anticipate in addressing this theme and how do you propose to tackle them? (10)**

1. Coordination between the physical implementation and animation. The 3-D model of pebbles has to be diminished when pebble is picked up and the level of water in pitcher has to be increased with every drop. It will be a challenge to check if pick-up or drop has been done and animate the models accordingly. This can be tackled by tracking the movement of crow-bot with help of aruco marker and making the bot efficient so that it doesn’t fail to pickup/drop pebbles.
2. Another challenge will be to reduce the traversal time. The bot is required to traverse at high speed but not miss its path doing so. This can be tackled by proper tuning of the white line sensor. Moving the bot at full speed will cause it to skit when stopped, this can be tackled by making duty-cycle 100% for a short duration in between the nodes and reducing it as node’s approached and also make it even lower while pickup/deposition.
3. One of the challenges will be to design a proper chassis for the bot. The size of bot should be large enough to contain all sensors and actuators but it should not collide with other things in the arena, specially the electromagnetic pickup mechanism, which needs to be proper in order to successfully place the pebbles inside the pickup/deposition zone without colliding with the markers. In order to achieve this, proper calculations regarding the bot’s structure is required. The chassis will be made octagonal in shape that is nearly circular making it small in size and at the same time, big enough to allot spaces for placing the hardware.
4. Encoders may return garbage values which will result in a challenge to maintain the bot’s alignment after turnings. This can be overcome by using quad-encoders and the white line sensor in tandem with each other.
5. The aruco markers of pick-up/deposition zones and of the crow-robot may get hidden from camera by structural parts of the robot while traversal like, robot’s body (if its height is too much) or the arm for lifting mechanism (if it’s made to fold and rise above height at which aruco marker is placed on the robot).

For this, it is ensured that the bot’s height is kept within a range and the arm’s design is such that it doesn’t come in between the camera and aruco-markers i-e the marker is always well exposed to the camera.