Algorithm for Development of Snake and Ladder Robot

[1]Chetan Pinto, [2]Chrystle Pinto
[1, 2]B.E. Student, Department of Electronics and Communication Engineering, SJEC, Mangaluru, India

Abstract

It is always great to have someone to ready to play at your will and give you company, be it for an adult or child. As it is being hard for humans to meet someone else's need robots are being implemented. While, there are AI based robots for chess and go, the simple game of snake and ladder has been over-looked. Proposed in this paper is a novel algorithm that can be implemented in a robot to play the game of snake and ladder.

Keywords: *Step increment, game piece, telescopic arm, swivel, move, identify*

I. INTRODUCTION

The game of snake and ladder needs no introduction. The die is rolled and the game piece is moved by the number of steps indicated by the die. This game has been recreational for people of all ages over the years. The game requires at least two players to declare the one who reaches the 100th step as the winner. The paper presents the algorithm for the robot that can give company to someone wanting to play this game. The implementation of chess playing robots [1] require competing strategy and high level processing while this game requires only predefined movements based on the logic and current results. The main system algorithm is proposed in section II. The possible way to identify the number from the die is presented in section III. Section IV and V provide an insight in the implementation of the hardware of this algorithm. VI section gives the conclusion and possible further development in this work.

II. ALGORITHM

Certain requirements before coming to the main system algorithms are to be initialized.

a) A counter to keep track of the movement and current position of the game piece.
b) A constant to store the length of the side of the square on the board
c) Four arrays
   LS = {x1, x2, x3. . . xn} to hold the numbers where the ladders start.
   LE = {y1, y2, y3. . . yn} to hold the numbers where the ladders end.
   SS = {w1, w2, w3. . . wn} to hold the numbers where the snakes start.
   SE = {z1, z2, z3. . . zn} to hold the numbers where the ladders end.

A given position in the array should correspond to starting and ending point of a particular ladder or snake. i.e. say i=2, in the array LS has a number where the ladder begins then the same position in LE must have the number where the ladder ends.

A. *Main Algorithm*

The Algorithm is depicted in flowchart shown in figure I. The system works by pushing the game piece from one step to another and the counter keeps track of the current location of the game piece by incrementing itself or updating as the case. Once the number has been retrieved from the die it is given to the microcontroller where it undergoes an iteration with count equal to the number on the die. In each iteration, it checks for the series beginning numbers i.e. 11, 21,31. . . and calls a function '*move'*. At the end of the iteration it has to check if the position is the mouth of the snake to descend or the base of the ladder to ascend. This is done by checking if the current position number exists in LS or SS array. If no, then no action is performed. If yes, then the position of that number is identified and the game piece is moved to the corresponding number in the same positon in LE or SE array. After reaching the final point it has to call another function *'identify'*. The reason to check for the series beginning number in each iteration is to differentiate between horizontal push and vertical push. Say, the game piece is to be moved from 9 to 10 then it is pushed horizontally. But if the piece is to be moved from 10 to 11, then it has to give a vertical push therefore the counter is monitored for such numbers.

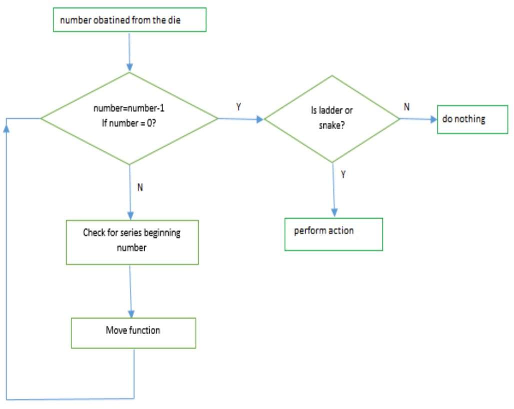

Figure I

B. *Function 'move'*

This function will have to perform two task. The first is to move the game piece to the next square. Second to move the chassis of the robot by a distance almost equal to the side of the square. The implementation of this is in section V

C. *Function 'identify'*

The need for the function arises while deciding if the game piece has to be moved horizontally left ways or right ways. This is done by classifying based on the digit in the ten's place. If the ten's place digits are even, then the coin is moved right ways if the digit is odd, then left ways. But this depends on the individual board whether the numbers start from left or right. This function takes the number of the current position into consideration. The function is called after the ascension of the ladder or descending down the snake to check in which direction is the next move.

III. NUMBER IDENTIFICATION

The system requires another arm mounted on a stand with a shoulder joint to throw the die and capture the image. Now it will be a daunting task to retrieve the number from the die thrown on the table. Hence to ease the work, the die is placed in the arm which has a cupped end (similar gesture as cupping the palm). A camera is mounted exactly vertically above the die in the cup to a horizontal stand in order to be able to capture the image. When a switch is pressed, the arm is quivered which will shake the die and when the quivering stops the die will be resting with a new face upward. The overhead camera captures the image of the die and compares it with the images in the data base. The data base must have images of all the six faces of the die clicked from an appropriate distance, almost equal to the distance of the camera and the die in the arm. The task of image comparison can be performed through the image processing toolbox in MATLAB as in [3], or through Python [2] or OpenCV. As the paper intends to present the working algorithm it has only put forth the ways to get the task done and does not endorse any method. But if Python is used then Mean Squared Error [4] method is suggested. The equation being:

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

Thus, it will be possible to identify the number and send the number to the microcontroller which has to handle the action. The mode of communication can be serial communication or I2C based on the features of the microcontroller.

IV. HARDWARE IMPLEMENTATION

A. The Arm

We have already loaded the length of each square on the board onto the microcontroller. The arm that moves the game piece is designed on a moveable chassis with a swivel base. This arm is different from the arm which throws the die which is fixed at the bottom. The size of the chassis is 12 x 7 inches or any other proportionate size. The swivel base is controlled with a servo motor and it supports a vertical arm onto which a telescopic horizontal arm is mounted. The end of the telescopic arm has two servo motors configured in such away, that one servo motor enables the game piece to be pushed sideways. The other motor turns the arm attached to it by right angles whenever there is a change in the direction of push. i.e. from horizontal to vertical. The vertical movement are enabled by the telescopic arm which moves front or back as per the need. All these movements are controlled by the microcontroller and selection of the microcontroller decides how fast the algorithm is implemented and a suitable action is performed.

B. Working

Initially the game piece is assumed to be placed at the centre of the first square and for every step it is pushed by a distance equal to the length of the square. At the row end the servo rotates and the piece is pushed vertically upward. In the next move the servo rotates again to change the direction of movement which will be opposite to the below row. When the ladder base or snake mouth is detected, the controller takes the current position and final position numbers from the array. From these the absolute value of the difference of the ten's place and unit's place is taken and the hypotenuse and the angle made by the hypotenuse with the horizontal is calculated. The swivel is rotated by the complement of the angle thus obtained. The direction of clockwise or anti-clock wise is determined by the following logic

Let,
d= difference between current and final position unit's digit.
ad= absolute difference between the current and final position ten's digit.

```
IF((d>0 & ad=odd) || (d<0 & ad=even))
        Move clockwise
ELSE
        Move anti-clockwise
```

V. NEED OF THE CHASIS

As already mentioned when the function move is called the chassis is moved by a distance equal to the length of the square. This is possible with the replacement of dc motor with stepper motor for the wheels. Moving the chassis along with each move keeps the arm in alignment with the current position of the game piece. The distance to be covered by the wheels is based on the area of contact of the wheels with the ground and the step size of the motor. This should movement must be in relation to the length of the square on the game board. Thus, the amount of turn for one increment in position is calculated and is used by multiplying with an integral number as needed.

A. When Encountering a Ladder

When a ladder is encountered, the game piece is pushed up the ladder by the telescopic arm in the direction turned by the swivel base. The chassis covers the horizontal distance from the current position to the new position based on the difference in the digit in unit's place. The decision whether to move forward or backward is decided the direction the swivel had turned.

B. When Encountering a Snake

The case is a little different when the snake is encountered. Using the same logic as in the case of ladder, the angle, the hypotenuse length, the direction of rotation is found out. But here the chassis first moves to align itself in the final positon and the pulls the game piece from its current position downward to final place along the hypotenuse. Thus, the piece reaching its required position after descent.

VI. CONCLUSION

The algorithm and technique proposed effectively covers the tracing of the path, ascending the ladders and descending down the snakes on any given board. However, it is not completely independent as the die is required to be placed in its cupped arm slot and a switch needs to be pressed for each move. Efforts can be made to eliminate this. Further development to this can be made by being able to move the game piece along the curve of the body of the snake which would seem more graceful to the onlooker.

REFERENCES

[1] Cynthia Matuszek et all, "Gambit: An Autonomous Chess Playing Robot Systems" 2011 IEEE International Conference on Robotics and Automation

[2] van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yarger N,Gouillart E, Yu T, the scikit-image contributors. (2014) scikit-image: image processing in Python. PeerJ 2:e453 https://doi.org/10.7717/peerj.452

[3] https://in.mathworks.com/help/images/ref

[4] Rosebrock , Adrian. How-To: Python Compare Two Images https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/