USER DOCUMENTATION

CS 101 EMBEDDED SYSTEMS PROJECT

OBJECT DETECTION IN UNKNOWN AREA
GROUP 264

Team Members:-

1. Pulkit Goyal 140070032
2. Parth Kothari 14D070019
3. Dhruti Shah 14D070002
4. Harsh Sheth 140040012

Contents

INTRODUCTION

Now-a-days, there are lots of applications where an autonomous bot is needed to track down a particular object in an unknown place and take subsequent actions. Or it is needed to scan a place inaccessible to humans such as in the army it can be used to detect whether an area is free of radioactive substances or not. And if such a substance is present, to know the exact location of the substance.

This thought has become the motivation to do the project. Here, we come up with system in which we add intelligence to “object-detecting bot” to scan an unknown arena and find a particular object.

PROJECT DESCRIPTION

The aim of the project is to design an autonomous bot which will avoid obstacles along its way and find out a specified object in an unknown arena. Even if the object is not present, it will scan almost the entire arena.

The object to be found out will be of a specific colour and shape and should be specified by the user.

The bot will traverse the arena scanning for the required object, avoiding any obstacle in its way. There will be a wireless communication between the bot and the laptop, to determine whether the required object is detected or not.

There will be a live feed from the webcam placed on the bot to detect the object, and make the bot respond accordingly if the object is detected.

REQUIREMENTS

A. HARDWARE REQUIREMENTS:

1. Firebird : Require 1 bot for detecting object
2. Web- Camera : Require 1 Web-Cam to click images of the arena at particular instants of time during its motion inside the arena.
3. Zigbee : To maintain communication between the bot and the laptop.
4. Proximity Sensors : To detect any obstacles in the path of the bot, to overcome them.
5. USB Cable : To connect the Web-Cam to the laptop

B. SOFTWARE REQUIREMENTS:

1. OpenCV : For processing image sent by the camera.
2. AVR Studio : To program information onto the firebird.
3. XCTU : For configuration of the zigbees.

USER INSTRUCTIONS

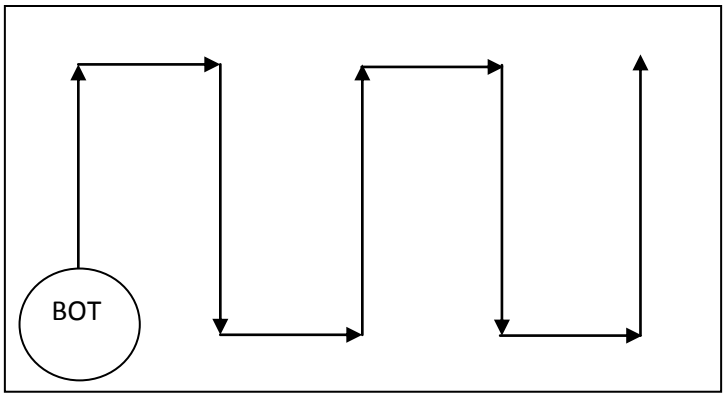
1. The user has to choose/specify the colour of the ball
2. The user has to provide the dimensions of the arena.
3. The user has to place the bot at the corner of the arena in the beginning.
4. The project can be implemented only on a WINDOWS system.
5. Since we are using a 64-bit compiler , the WINDOWS OS should be of 64 bits.

IMPLEMENTATION

FUNCTIONALITY:

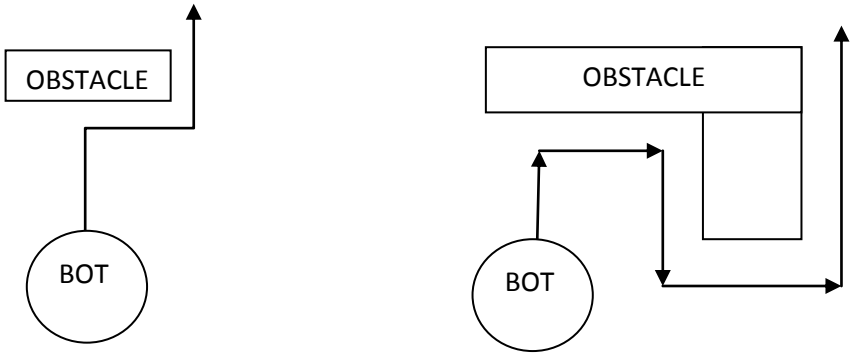
A. DETECTING THE OBJECT:

A. **MOTION OF THE BOT:** The bot will move inside the arena in a systematic manner (as shown in figure(i)). The x, y coordinates of the bot will be recorded at particular instants of time.



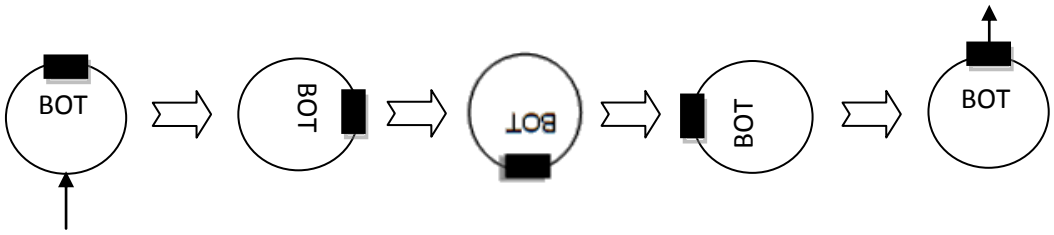
Fig(i)

B. **AVOIDING OBSTACLES:** If any obstacle comes in between the motion of the bot, the bot will avoid it and come back to its original orientation (as shown in fig(ii)). Attempts have been made to accommodate for as many types of obstacles as possible.



Fig(ii)

C. **LOOKING FOR THE OBJECT:** Snapshots of the arena are taken at regular intervals from multiple angles and positions to look for the object using a webcam connected to the laptop (see fig(iii)).These snapshots are then processed to search for the object with the help of OpenCV. If the object is detected , the bot will move towards it, else it will continue its motion looking for it.

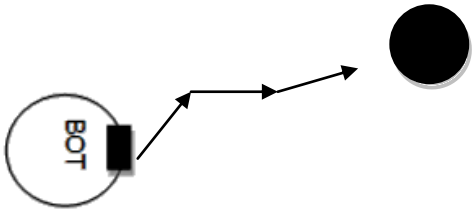


Fig(iii)

D. **COMMUNICATION BETWEEN THE BOT AND THE LAPTOP:** The communication between the laptop and the bot will be done with the help of Zigbees.

B. MOTION TOWARDS THE OBJECT:

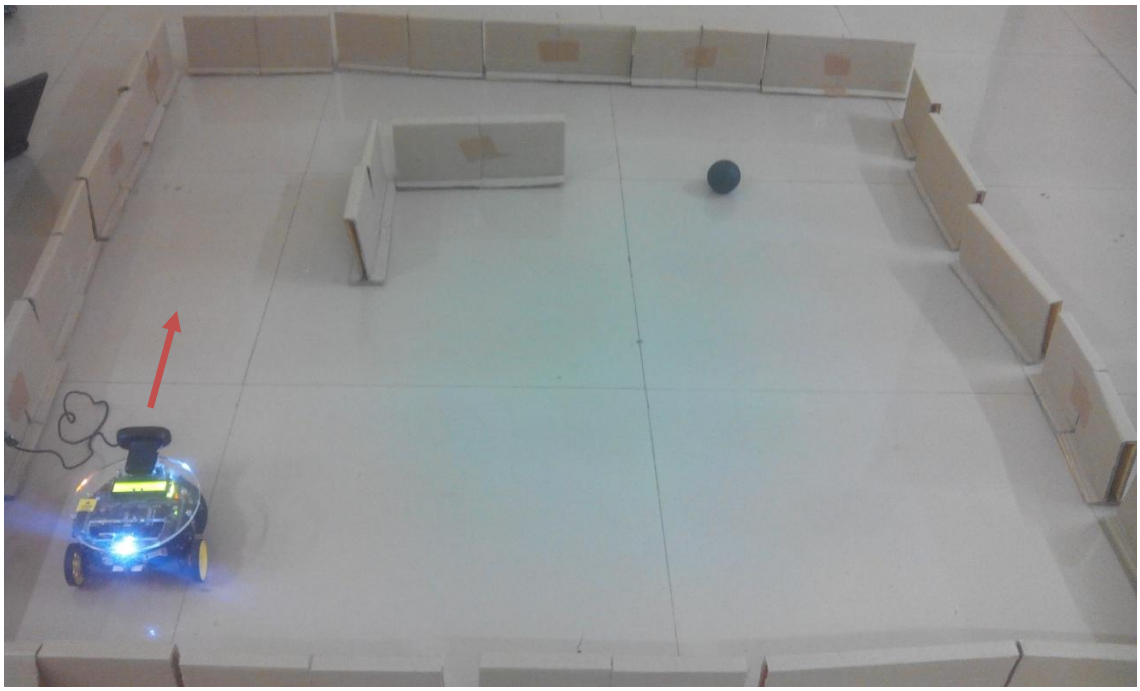
Once the object is detected, the laptop will send the distance between the centre of the object in the image and the centre of the image (as shown in the figure (iv)) to the bot. The bot will then rotate by a specific angle so that centre of the object coincides with the centre of the image and will then move towards it by a specific distance. The above process is then repeated until the bot is at some specified distance from the object.



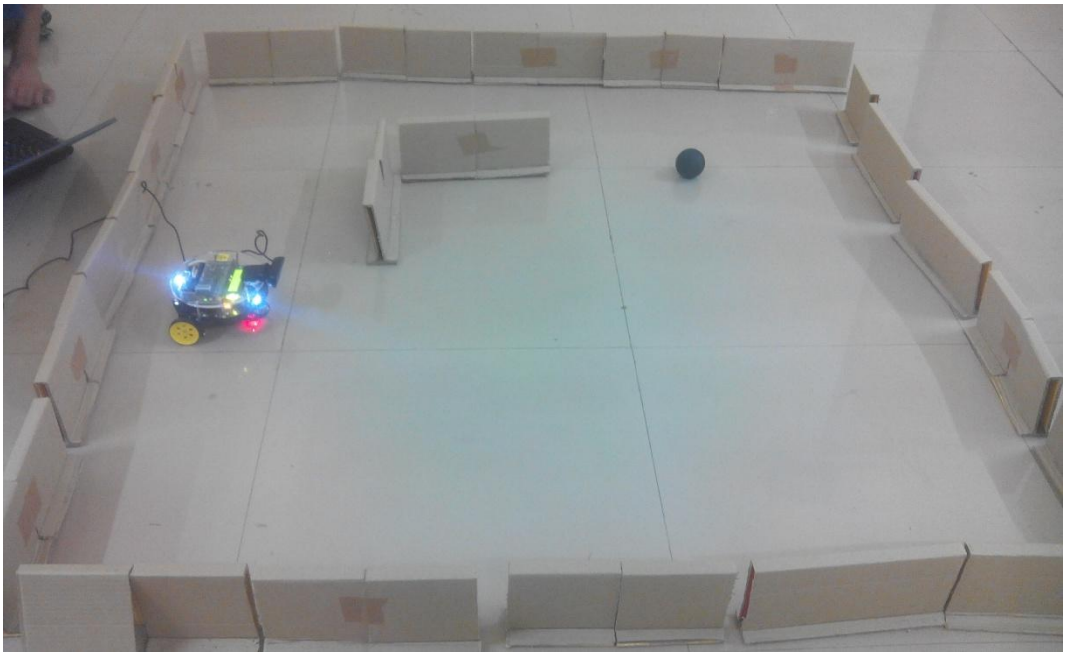
Fig(iv)

TESTING STRATEGY AND DATA

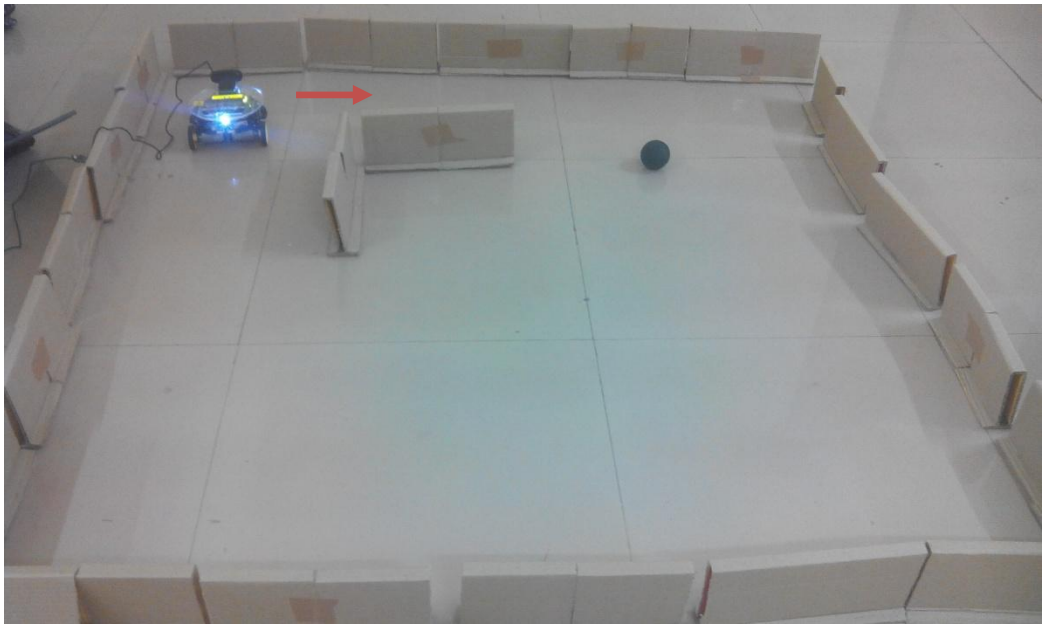
For testing the code, we have made an arena out of cardboard. The bot will start from the corner of the arena and will follow the algorithm which we have shown below in a series of images. They explain the algorithm the bot will follow while moving inside the arena.



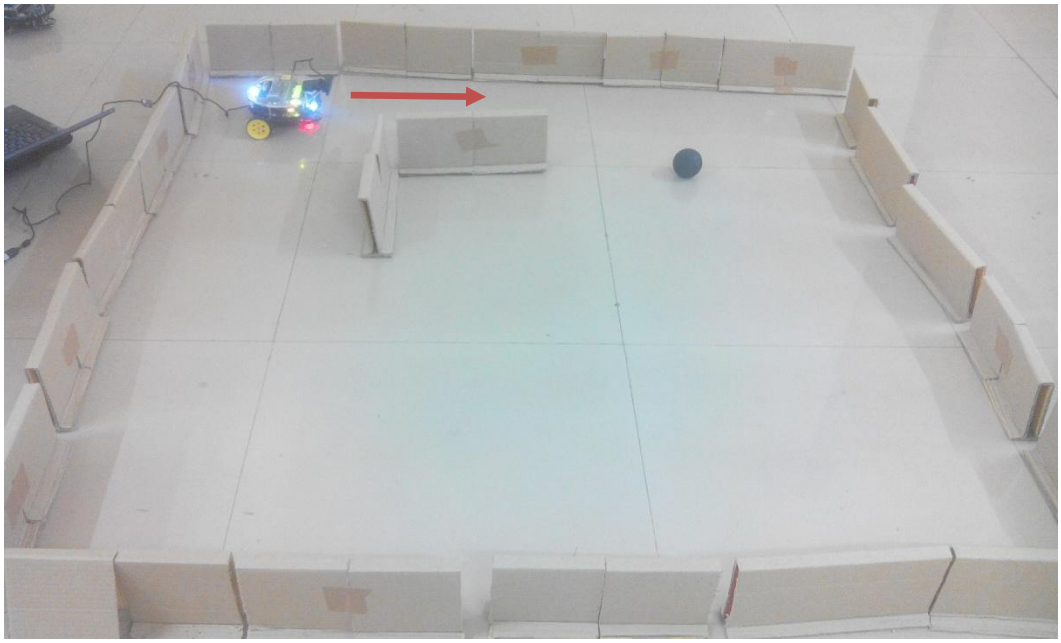
(i) The bot is at the starting position which is a corner of the arena initially. From here it will start moving forward and stop after a specific distance.



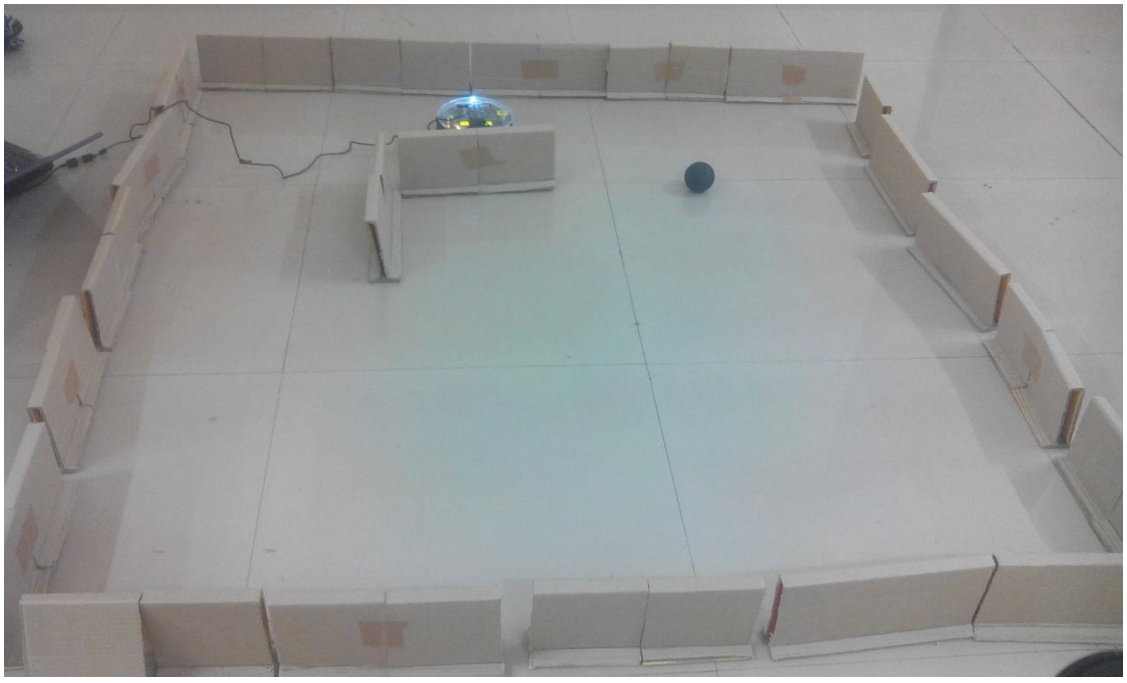
(ii) After moving ahead by this certain distance, the bot will rotate by 360 degrees (90 degrees at a time) to scan the arena for the ball. If the ball is not found, then it will continue its motion (fig(iii))



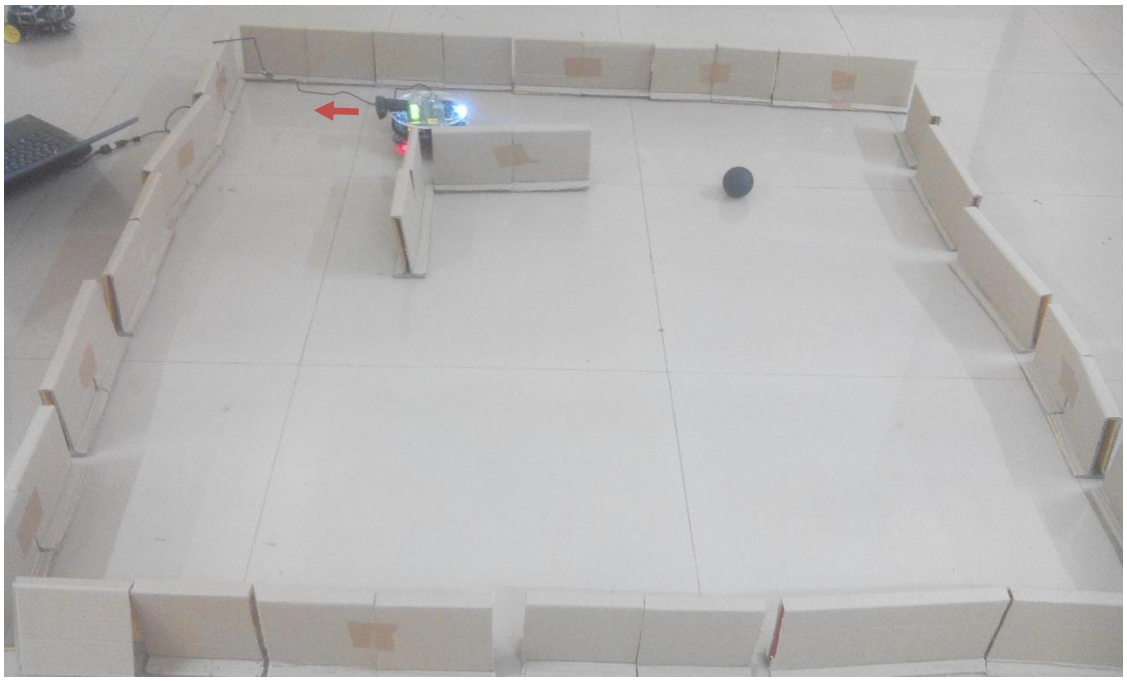
(iii) As soon the bot has moved forward by a distance equal to the arena length (given by the user) it will then rotate by 90 degrees and continue its motion along the arena's width.



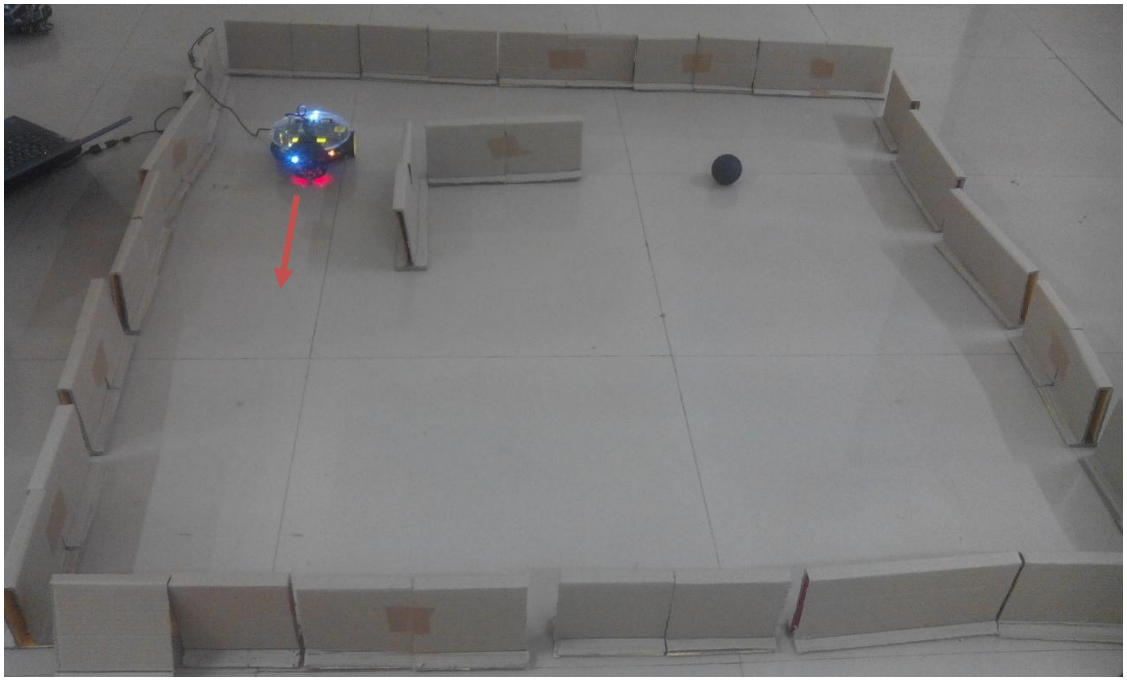
(iv) The bot will now move along the width of the arena for a certain distance, then take a right turn and continue its motion.



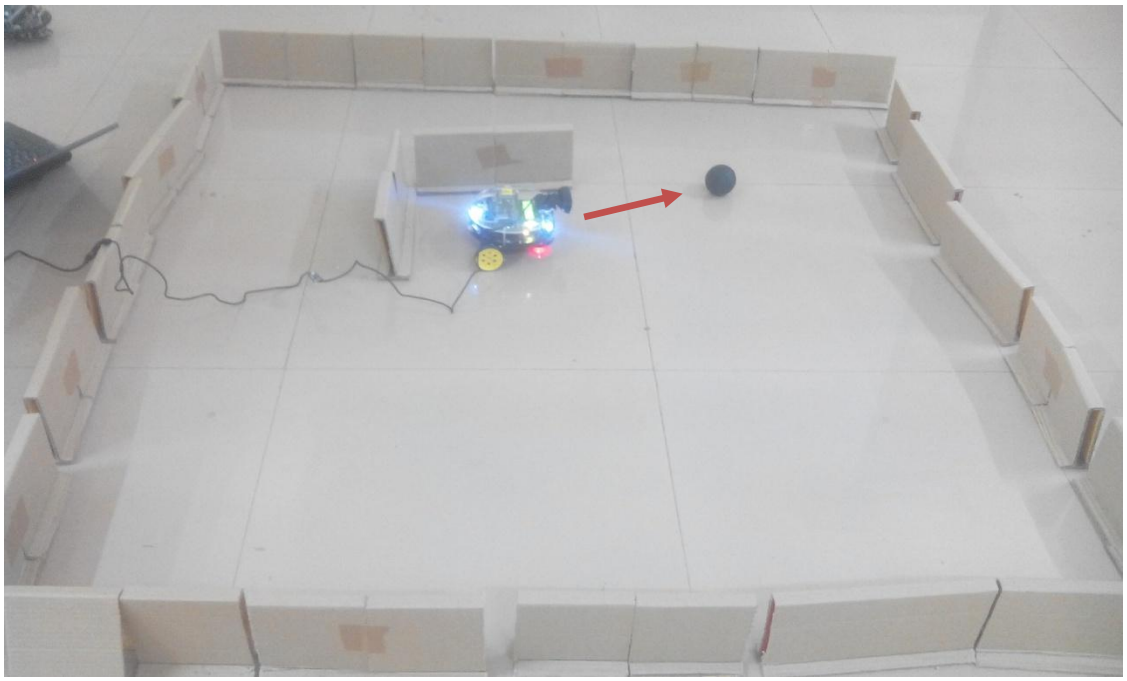
(v) If an obstacle is detected in the way of the bot, it will stop in front of it and scan for the ball. If not found, it will take a right to avoid the obstacle.



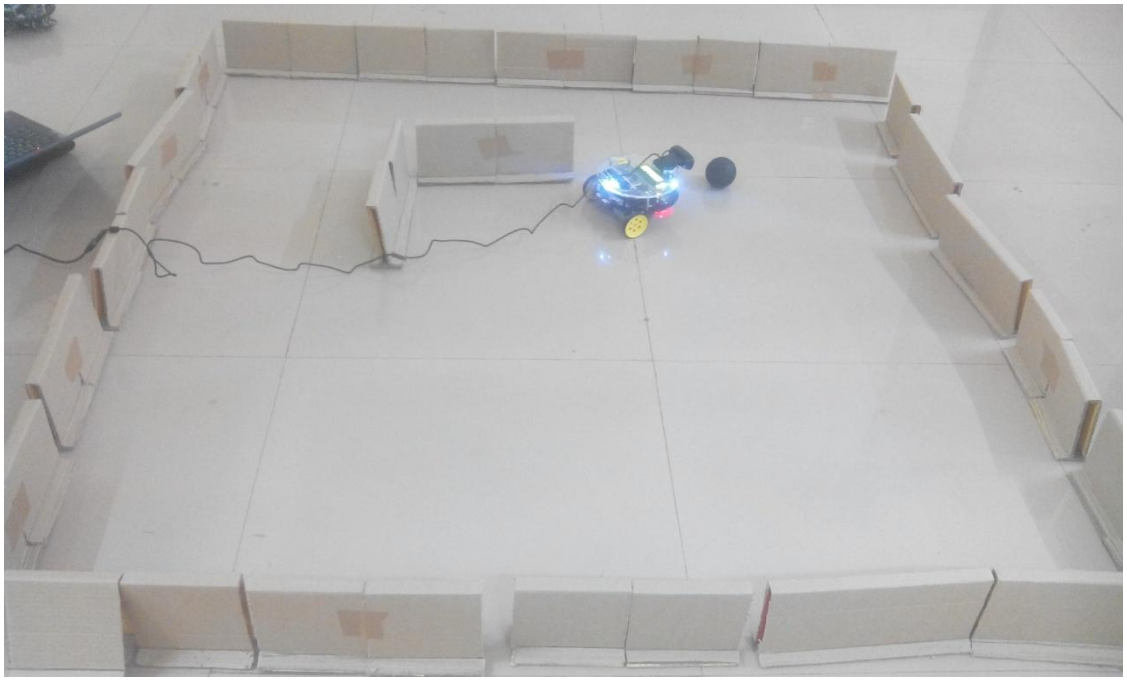
(vi) The bot moves forward and then again takes a right so as to restore its initial orientation.



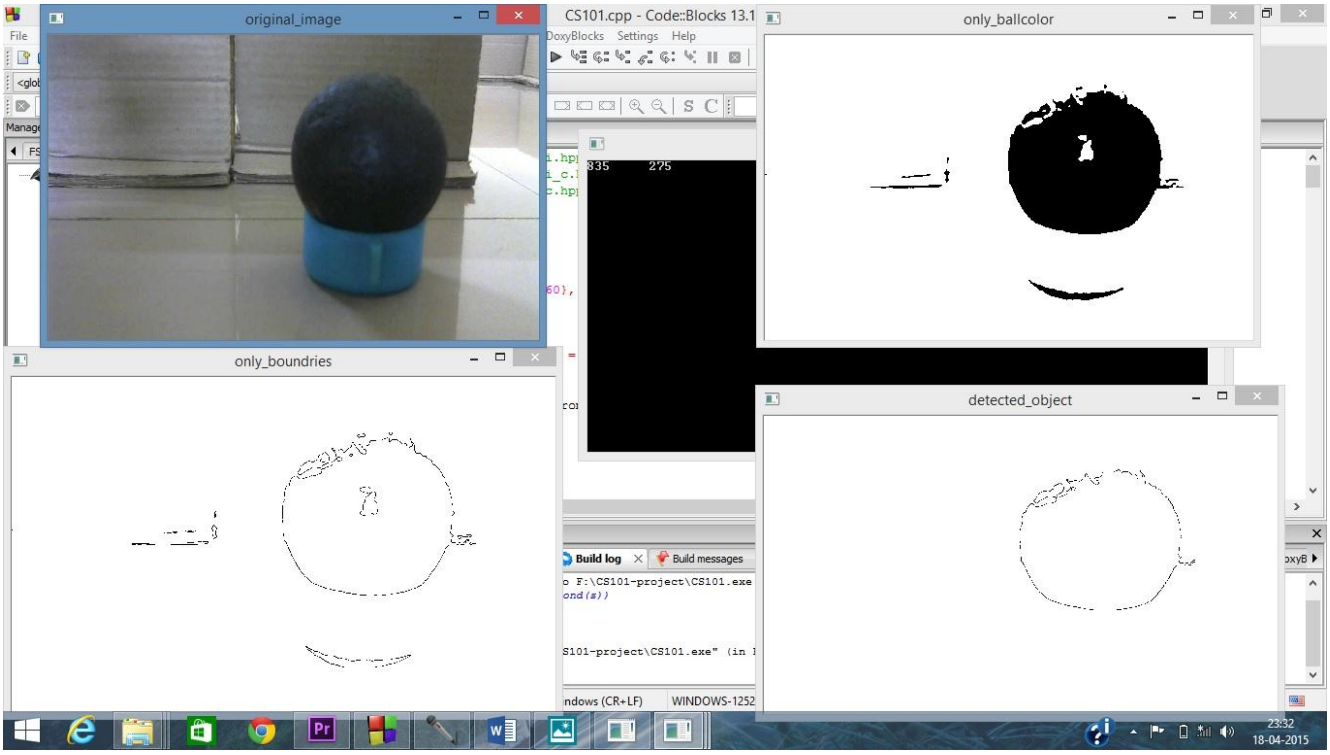
(vii) Once its initial orientation is restored, it will again continue its normal motion.



(viii) It will rotate 360 degrees at various intervals as in image(ii) to scan the arena for the ball.



(ix) Once the ball is detected, the bot will rotate till the centre of the ball coincides with the centre of the image.Then the bot will move towards the ball(as in fig(iv)) and stops at a specific distance in front of it.



(x) This is the screenshot of the laptop when the camera detects the ball.

(i) The window on the top left shows the actual image which the web-cam captures i.e. the original image.

(ii) The top right window named “only_ballcolor” whitens all those pixels of the original image whose colour is not the same as the colour of the object which was specified by the user. While the pixels of the original image whose colour matches with the colour specified by the user are made black

(iii) The bottom left window named “only_boundries” shows only the boundaries of the black-coloured objects which were obtained in the window “only_ballcolor”

(iv) The bottom right window named “detected_object” selects only those objects from the window “only_boundaries” which are almost of the same shape as specified by the user (the shape which we have inputted is circular) As you can see the image processing works just perfectly!

(v) Finally, the black window at the centre shows two numbers. These numbers are the x, y coordinates of the centre of the object in bottom right window. The distance of the x coordinate of the object from centre of image is then obtained, the bot is rotated correspondingly and then it moves towards the object.

DISCUSSION OF SYSTEM

A. What has worked according to the plan:

1. MOTION OF THE BOT:
   The algorithm of the bot was pretty successful in making the bot traverse inside the arena in a systematic manner. But a very crucial factor in this process is the hardware limitation. The bot should rotate exactly by the specified angle else the code may lose its accuracy. We tried to overcome this limitation by constantly calibrating the bot.

2. OVERCOMING THE OBSTACLES:
   The bot successfully overcame the obstacles in its way, maintaining its orientation.

3. ZIGBEE COMMUNICATION:
   Took a lot of time, but we were finally successful in establishing two-way communication between the bot and the laptop. This was a very important factor for the working of the whole project since image processing could not be done on the bot and data received from image processing was crucial for the working of the bot.

4. IMAGE PROCESSING:
   This section was pretty much what the project was based on. It was vital to be able to detect the object in an arena in real-time. We have managed to detect balls of different colours using a webcam mounted on the bot. The laptop searches for a ball in the images of the arena taken by the bot and sends the relative co-ordinates of the centre of the ball to the bot once the ball is detected.

5. COORDINATES OF THE BOT:
   We were successful in determining the x, y coordinates of the bot at different instants of time. This was a great help in tracking the position of the bot.

6. INTEGRATING ALL THE COMPONENTS:
   Integrating all the components (i.e. the motion, Zigbee communication and the image processing) of our project together was the most important part of our project. But we were successful in achieving what we had aimed for at the start!

B. What we added more than discussed in SRS:

1. THE RETRACE FUNCTION:
   There may happen a case when the bot moves towards an object , but it later turns out to be some other object (due to limitations of image processing) i.e. not the required object, then we have included a function called “retrace back” which will make the bot traverse back to the position from where that particular object was first spotted. Then, the bot will again continue its motion from there to find the required object.

2. CHECKS ON ALL COMMUNICATIONS:
   Once any data is sent from either the laptop or the bot, it waits for a particular signal to be received from the other end before proceeding in its code. It keeps resending the data until the signal is received. This mechanism was incorporated into the code to fix the problem of data loss in communication over long distances.

3. USED DOXYGEN :
   The final code, along with the comments, was documented using doxygen and converted to pdf format.

C. Changes made in plan:

1. USED WINDOWS.H INSTEAD OF LIBXBEE:
   We had decided to communicate to the libxbee library to communicate with the laptop’s serial ports directly from the C++ code. But due to various factors and after a lot of fruitless efforts, we finally had to look for other ways to go about our task. We have finally used windows.h, a C++ Header file for Windows systems, to complete our task. Windows.h had just the right pre-defined functions and structures required to accomplish all our set goals.

2. MOTION ALGORITHM:
   We had to change our motion algorithm. Initially, we were going to move along the boundary of the arena. According to the new algorithm, we will move according to fig(i) shown above.
   Cause:
   The efficiency of scanning the arena was much greater according to the new code.

PROBLEMS FACED AND THEIR SOLUTIONS

We had faced a great number of challenges at every point in our project, some of which are listed below:-

1. The first one was attempting to set up enable the code to automatically initiate communication between the laptop and the bot. This issue was resolved using the header file windows.h.

2. Hardware limitation – The bot does not rotate through the exact angle specified despite several attempts at calibration. To overcome this we kept on calibrating, until we got the desired result

3. Image Processing may fail to detect the object in case of insufficient lighting or due to lack of colour difference. To overcome this, we kept sufficient lighting while carrying out test cases as well as ensured there was a proper colour difference between the object and its surroundings.

4. Another problem about image processing that we faced was that often gave incorrect results i.e. it identified incorrect objects as our required target. We made provisions for this by making the bot retrace its path every time it realised it was moving towards an incorrect target.

FUTURE WORK

1. The bot can be used to locate a variety of objects in any possible area. This gives it a wide range of applications ranging from locating missing objects to detecting radioactive substances or bombs.

2. It can also be used to search and traverse the terrain inaccessible to humans using just the motion algorithm of the bot.

3. The bot can be programmed and modified to carry out a number of tasks, after the object is detected like a tennis ball collector or a garbage collector etc.

4. Right now, we are using 1 sharp sensor for object detection. We can increase their number upto 5 for much efficient object detection.

CONCLUSION

This bot acts as a precursor to a wide range of applications in numerous important fields, since the bot can be programmed and modified to do various tasks once the desired object is detected. It can be especially useful for exploration and probing purposes.

REFERENCES

1. www.digi.com/xbee
2. ATmega2560 Datasheet by Atmel
3. http://opencv.willowgarage.com/documentation/cpp
4. https://zahidhasan.wordpress.com/2013/02/16/how-to-install-opencv-on-windows-7-64bit-using-mingw-64-and-codeblocks/
5. http://www.opencv-srf.blogspot.in
6. Firebird V Hardware and Software manuals
7. E-Yantra previous projects and tutorials
8. Project Video: https://www.youtube.com/watch?v=V70DxupY36g
9. Screencast: https://www.youtube.com/watch?v=22fVzaRpfMI
10. Github repository: https://github.com/Luffykun/ObjectDetectionInUnknownArena