Course Project Documentation

<u>CS101</u>

Arena Protector

Team ID: - 406

Ayush Agrawal	**: 14D070042**
Mohak Goyal	**: 140070029**
Raman Preet Singh	**: 140070059**

Table Of Contents

1. Introduction:

The project, made on embedded C using Firebird Atmega 2560, aims at identifying objects with their colour and then carrying out some specific function. The vision behind the project is to protect an area from invaders and trespassers. The bot inspect the arena via a camera mounted upon it. As soon as it identifies the object specified, it will move towards it and push it out of the arena. The bot does not push objects that are already out of the arena.

2. Problem Statement:

The aim of the project is to design an autonomous robot which will identify and push objects of red colour (attribute taken for this project) out of the arena bound by a black boundary line. Arena will be a plain restricted area. Objects should be stationary within the arena. Goal is to identify the object of red colour among the various objects in the arena, push it out of the arena and then come back into the arena to continue the procedure for other objects.

3. Requirements:

A) Hardware Requirements

1. **FireBird**: Required 1 bot.
2. **Camera**: To capture images of the arena.
3. **Xbee**: To maintain communication between bot and system.
4. **Proximity Sensor**: To measure the distance between the bot and the object.

B) Software Requirements

1. **OpenCV**: For processing images sent by camera
2. **AVR studio**: To program instruction onto a given bot
3. **XCTU**: To configure Zig-bee.

4. Implementation:

A) Functionality:

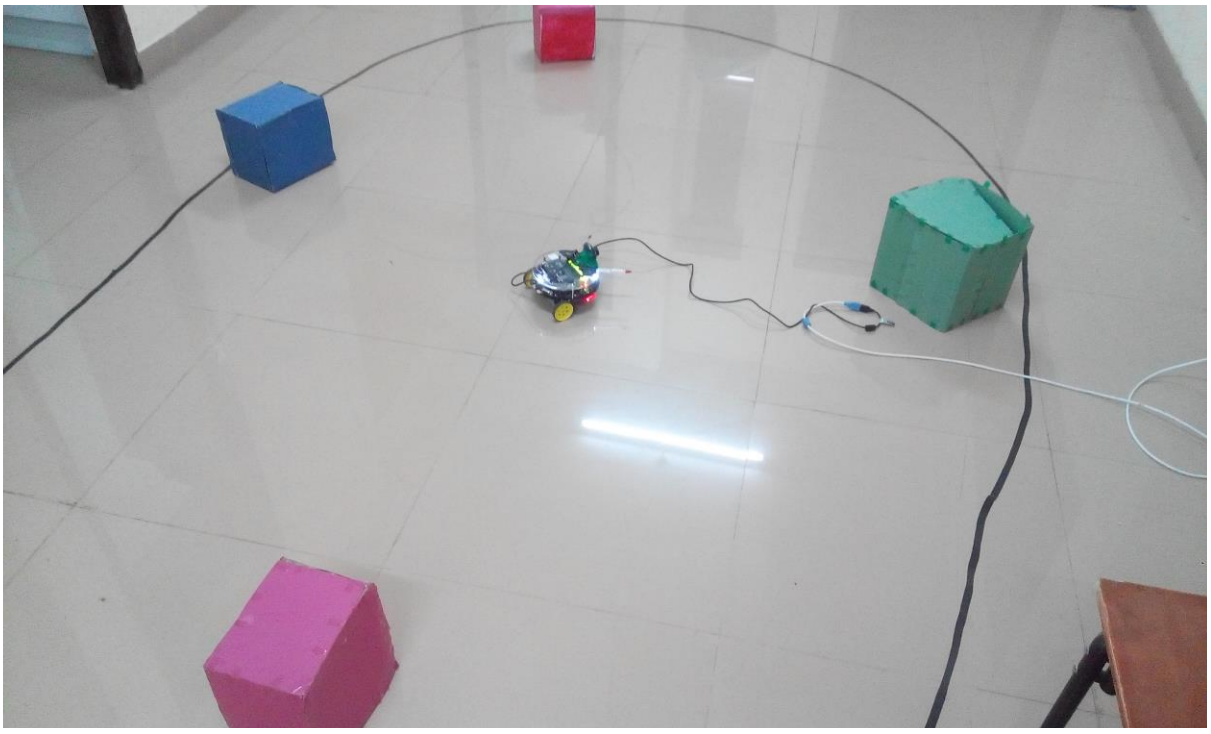
a) Determining position of object: The bot will make rotations by a fixed number of degrees. After each such rotation, it will take photograph of the portion of arena in view. This photograph will be processed and tested for the presence of required object. Affirmation or denial will be sent to the bot via Xbee. The bot will continue with the next rotation if denial is received. In the other case, i.e., when affirmation is received, it will break from the loop of rotations and start advancing in that direction.

b) Pushing the recognised object out: After recognising the object, the bot starts advancing in the direction of the object. For confirmation, it stops when it comes within the proximity of any object (distance less than 20 cm). This is just to ensure that it doesn't bump into any other object in its way. This also ensures that no other object has been placed in its way. Now it again asks for image processing. If confirmation of red coloured object is obtained, it continues, pushing the object out of the arena. In case confirmation of red object is not obtained, it rotated by 90 degrees and continues the initial procedure from a different orientation. A mechanical setup is made on the bot to ensure that that object cannot slip out of its grip while pushing.

c) Returning back to arena after pushing out the object to a certain distance: while pushing the object, the bot comes over the boundary, i.e., a black line. The line sensors note this. From this instant the bot continues to push the object through 1m distance. After this distance is reached, the bot continues to retrace its path, leaving the object outside. The bot enters the arena through some distance and orients itself randomly to continue detecting another red object. One salient feature of the bot is that it would not cross the black line boundary without an object. If it comes across the boundary while detecting the objects, it would turn back.

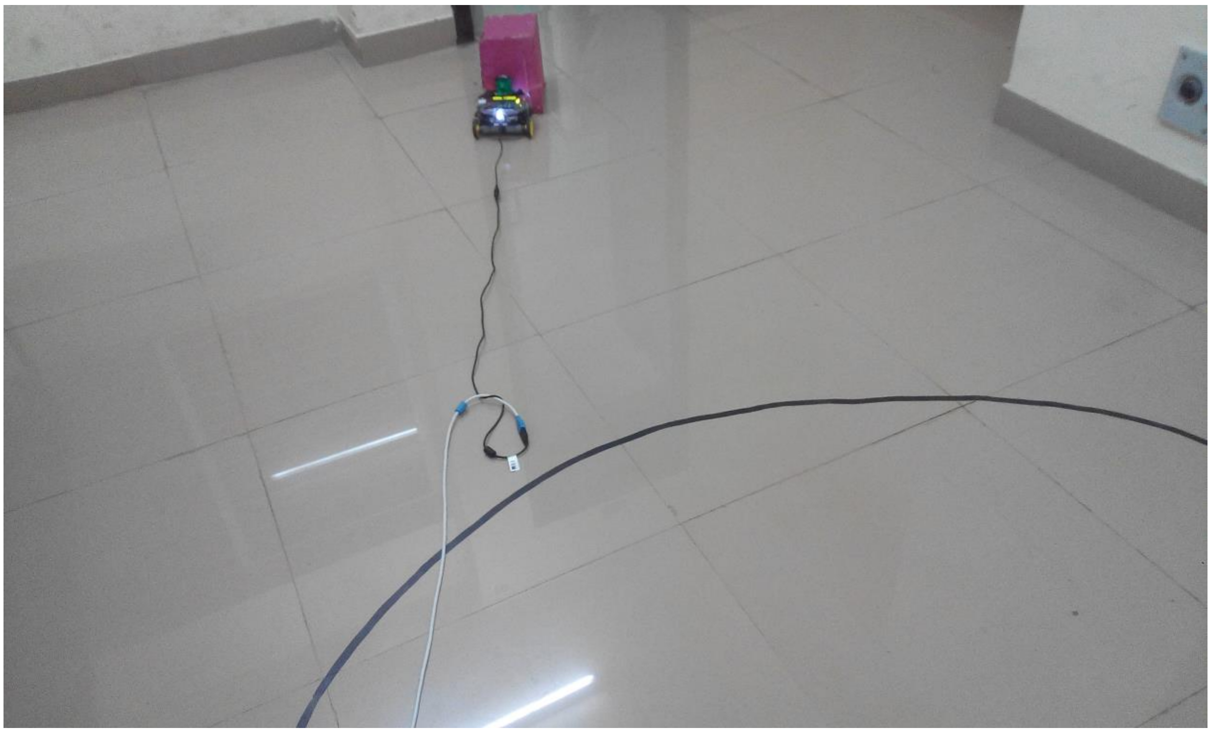
5. Testing Strategy and Data:

Bot in initial orientation:-

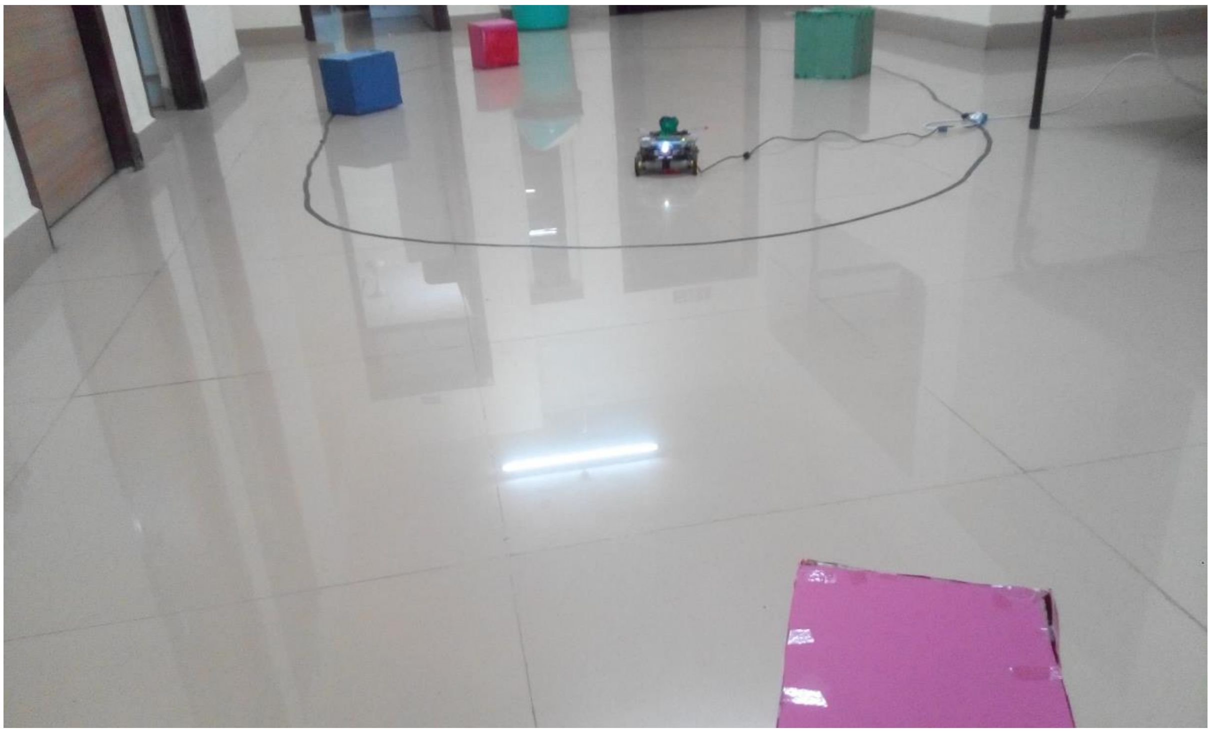


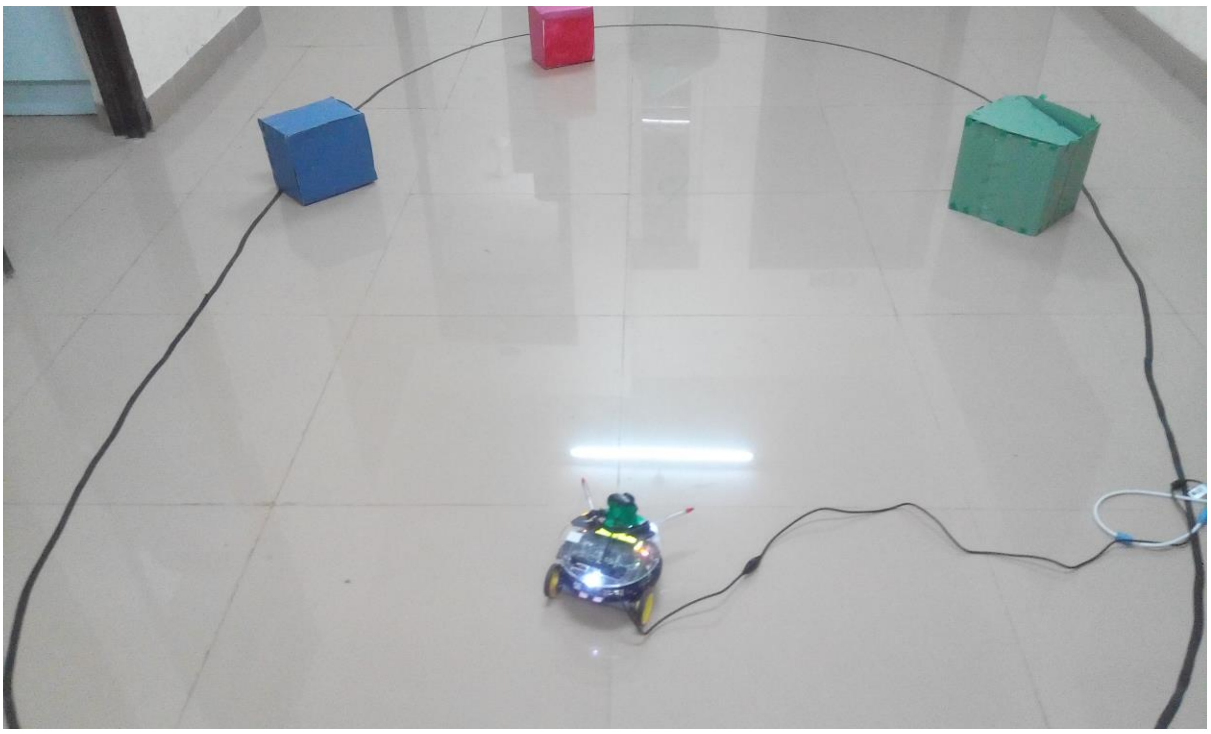
Bot oriented facing red object:-

Bot pushing object out of arena:-



Bot back to the arena:-





6. Discussion of System:

A) **What things worked as per plan**?

1. Object Identification by successful image processing algorithm.

2. Successful pushing out of object and return of bot in the arena.

4. Zigbee Communication: Successfully able to communicate with the required bot by using Zigbee and able to take appropriate action at real time.

B) **What we added more than discussed in SRS**?

1) Previous action based control: In-order to avoid redundant rotation of attacker bot when the marker is not visible in its viewing area we implemented previous action based controlling of bot i.e. when the marker is not visible on screen then attacker bot will rotate in the same direction of the previous.

2) Obstacle Detection and Avoidance: We are able to successfully distinguish between obstacle and our target object and avoid pushing or even touching bodies that are not of red colour.

(C) **Changes made in plan**:

1. Initially we had planned to use wireless camera and send pictures to the system via Wi-Fi, but due to unavailability of the hardware, USB camera was used.

7. Future Work:

A) In our present project we worked only with red objects. In the future it can be extended to various attributes of the object. The attribute to be considered can be taken as input from user.

B) The bot operates only in well-lit rooms. Its functionality can be extended to dark rooms by installing torch on it which would operate as per the illumination in the room.

C) The functionality of the bot can be extended to eliminating movable objects that can try to run away from the bot. For this the motion of the object has to be tracked by the bot and its trajectory should change accordingly.

D) The gripping mechanism of the bot can be improved so that even if the object tries to get free of it, it cannot.

8. Conclusions:

At last we want to conclude the CS101 Embedded Systems Project with an emotion of relief, satisfaction and enjoyment.

By doing this project we got to learn about embedded systems' coding, using various sensors, communication between two devices and most importantly (as this was the most difficult part) Image Processing.

Also we would like to thank our Professor Kavi Arya Sir for giving us the opportunity to work on this project. And we would also like to thank all the TAs and friends who helped and guided us to complete this project.

9. References:

1) OpenCV Installation Guide: “http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/”

2) Embedded Systems Coding and Xbee configuration:

eYRC tutorials.

10. Links for various softwares to be downloaded

1) OpenCV: “http://opencv.org/downloads.html”

2) MinGW: “http://sourceforge.net/projects/mingw/files/”

3) CMake: “http://www.cmake.org/download/”

4) CodeBlocks: “http://www.codeblocks.org/downloads/binaries”

5) XCTU: “http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu”

6) GitHub Repository Link: “https://github.com/ayushagrawal/BasicControls.git”

7) Video of working bot: “http://youtu.be/MpA2rqlZaOg”

8) Video of installing softwares: “https://youtu.be/2tSyxbrJBDM”