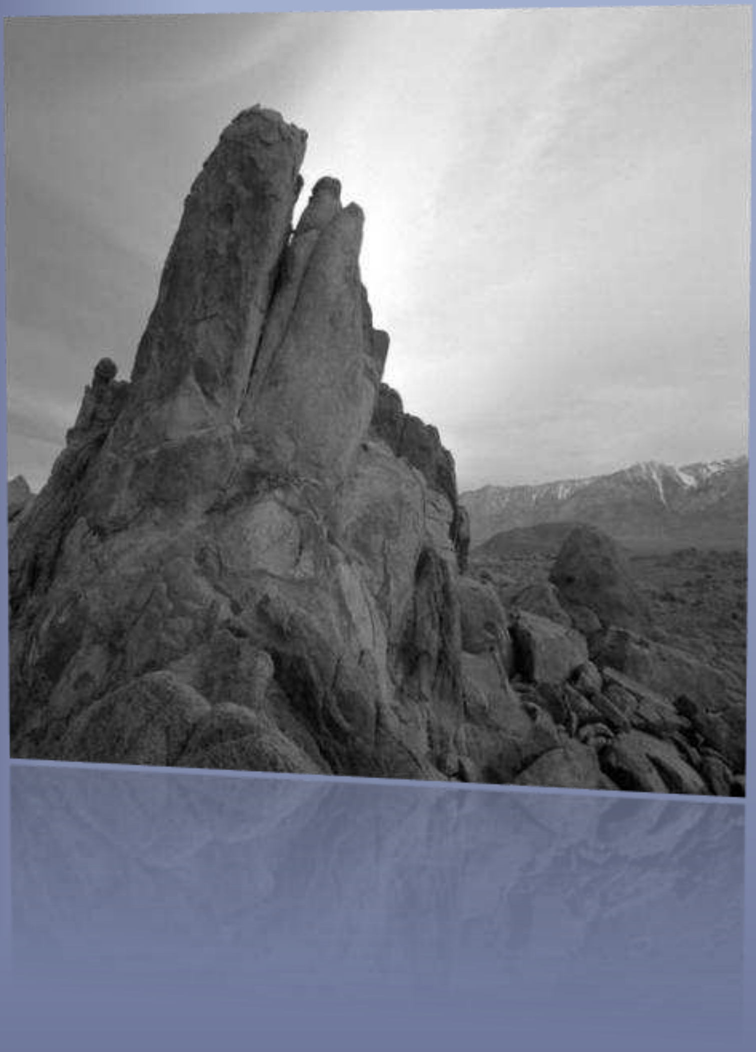


MAP TRACER

CS684 – 2010 Project Documentation

Exploration of an unknown environment searching operations inside buildings, caves, tunnels and mines are sometimes extremely dangerous activities. The use of autonomous bots to perform such tasks in complex environments will reduce the risk of these missions.

Contents

Introduction

1.1 Document Objective

The objective of this document is to help someone else run the code that is delivered as part of this project.

Project Title: Map Tracer

Students:

Name	Roll No.	Email
Arpit Malani	10305901	malani@cse.iitb.ac.in
Hermesh Gupta	10305080	hermesh@cse.iitb.ac.in
Rahul Nihalani	10305003	rahuln@cse.iitb.ac.in
Vivek V Velankar	10305050	vivekvelankar@cse.iitb.ac.in

1.2 Project Objective

Exploration of an unknown environment searching operations inside buildings, caves, tunnels and mines are sometimes extremely dangerous activities. The use of autonomous bots to perform such tasks in complex environments will reduce the risk of these missions.

1.3 Goals

Generate a topological map which reflects the map of unknown arena. As a result, this method needs significantly less time to accomplish the task.

1.4 Working Model

Our method includes three main tasks: navigation and exploration, localization.

1. Navigation : Navigate the Bot according to physical dimensions of unknown territory. Decision making for Bot in an unknown environment is a very complex problem. Since nobody knows what is after the frontier of an

unexplored area, there is no unique optimum algorithm that is completely reliable. In such situation a Bot should make a decision to progress exploring objects based on IR sensors and proximity sensors values without collision with boundary walls or any other object in the field.

2. Exploration : Exploring the unknown spaces and obstacles using Sharp IR sensors and proximity sensors. There are basically two proposed methods to exploration-
    a. Explore empty space
    b. Explore solid obstacles

   What we implemented was to make the bot traverse the whole arena without hitting the walls or in-between obstacles. For this we defined three type of threshold distances:-
    a. THRESHOLD - This denotes the distance that must be available to travel straight while travelling in free areas.
    b. MINITHRESHOLD - This distance is the distance that must be available on sides to go straight otherwise turn on a side that has more free area.
    c. SHEERTHRESHOLD - This distance is used as threshold when we are exploring a small and irregular arena.

   In the same way we defined three types of delays for different types of distances available to robot for exploration. Using these we developed an algorithm which checks the value of sensor values against the thresholds and takes the necessary action.

3. Localization : Localizing Bot position by implementing local GPS and basic image processing. **Simultaneous localization and mapping** (**SLAM**) is a technique used by robots and autonomous vehicles to build up a map within an unknown environment (without apriori knowledge) or to update a map within a known environment (with apriori knowledge from a given map) while at the same time keeping track of their current location.

   To implement localization we have implement local GPS to track the position of the Bot.

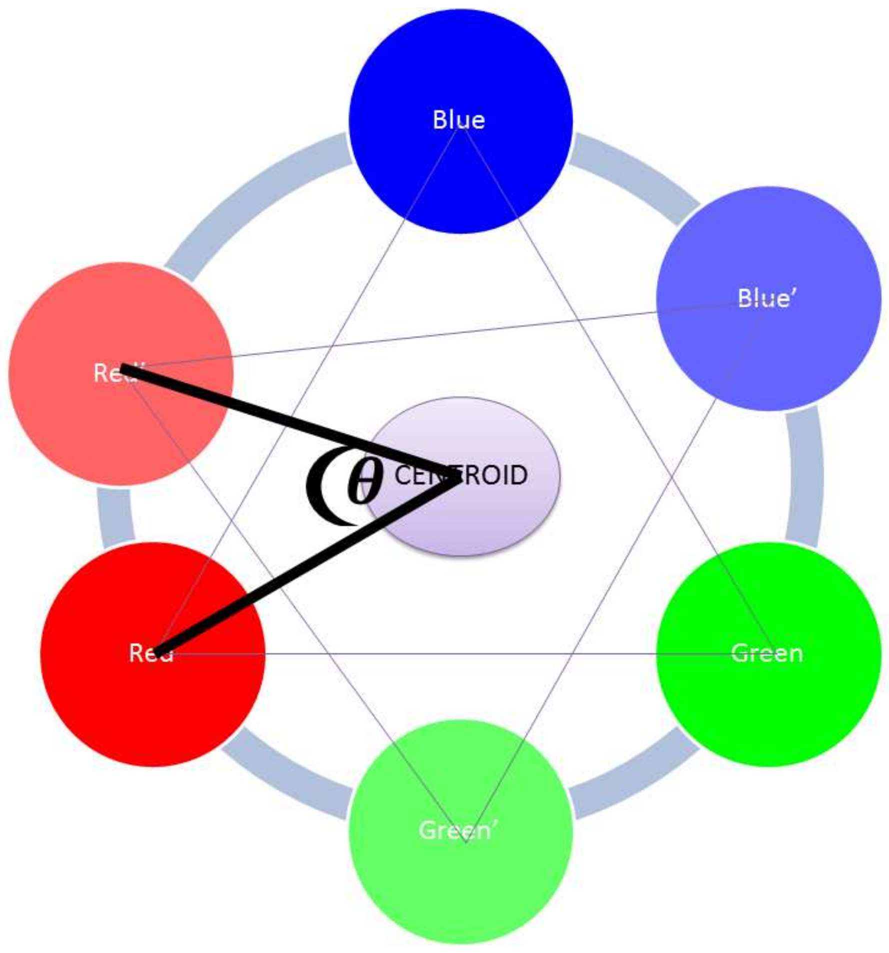


Figure-1 Local GPS Implementation

Local GPS System Working Model (fig.1)

I. Take Red, Green and Blue circles and put them in at the vertices of the equilateral triangle

II. Mount this triangle at the roof of the unknown environment whose map we want to trace

III. A camera is mounted on bot to capture the real time image of the triangle

IV. Then by image processing on captured image we can calculate the centroid and angle of the triangle

V. Then use this centroid and theta to interpolate the position of Bot in world coordinate system

VI. These points are used to draw map

Design

2.1 Architecture Diagram

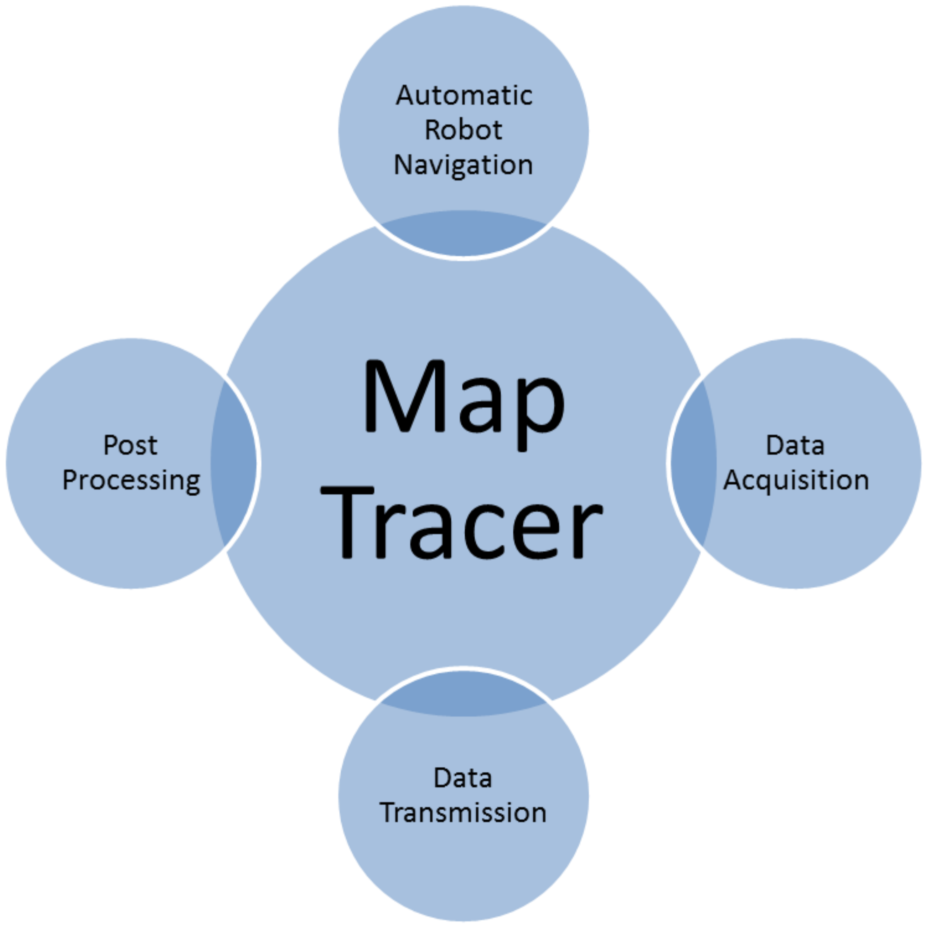


Figure-2 Architecture Diagram

2.2 State chart

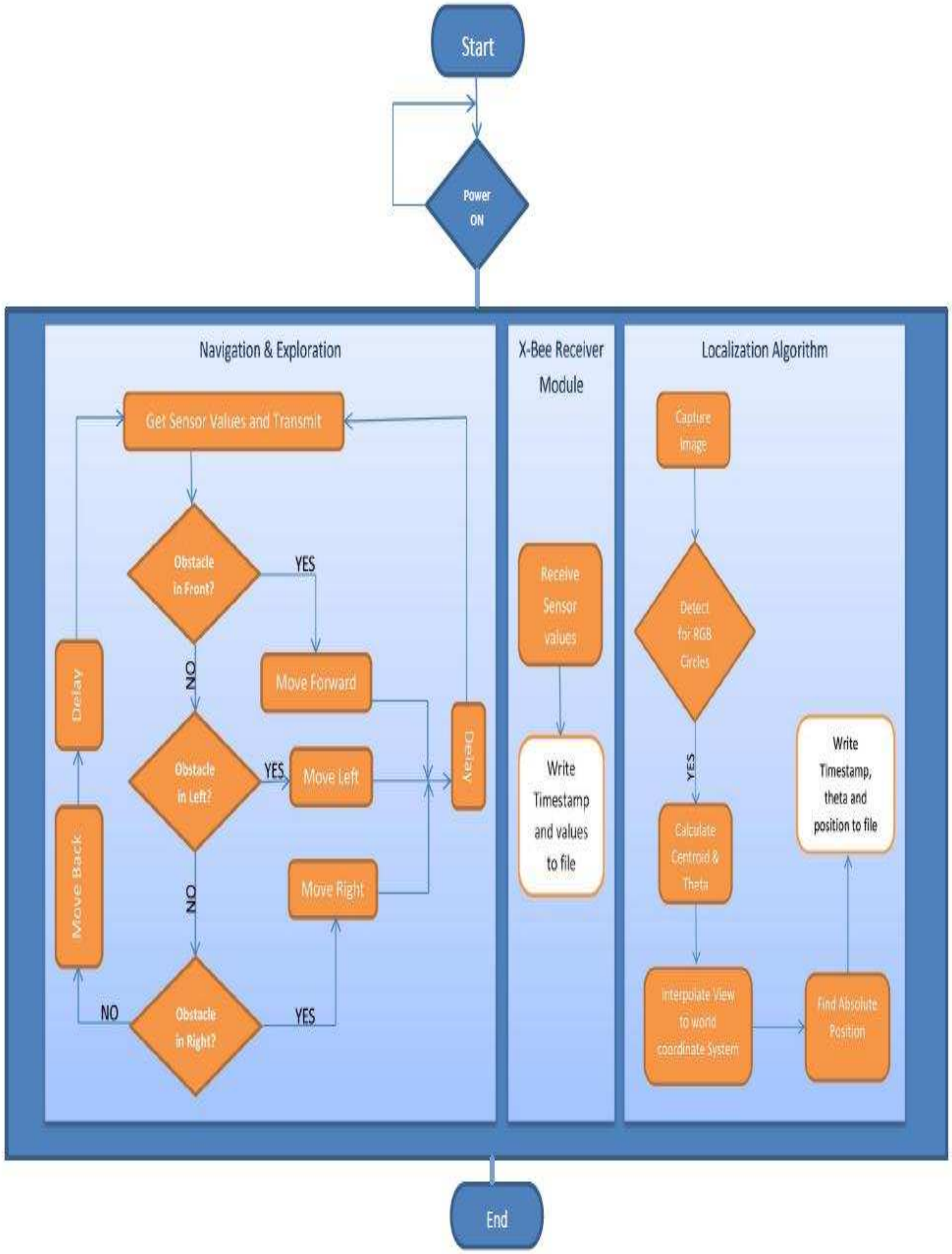


Figure-3 State Chart for Pre-processing

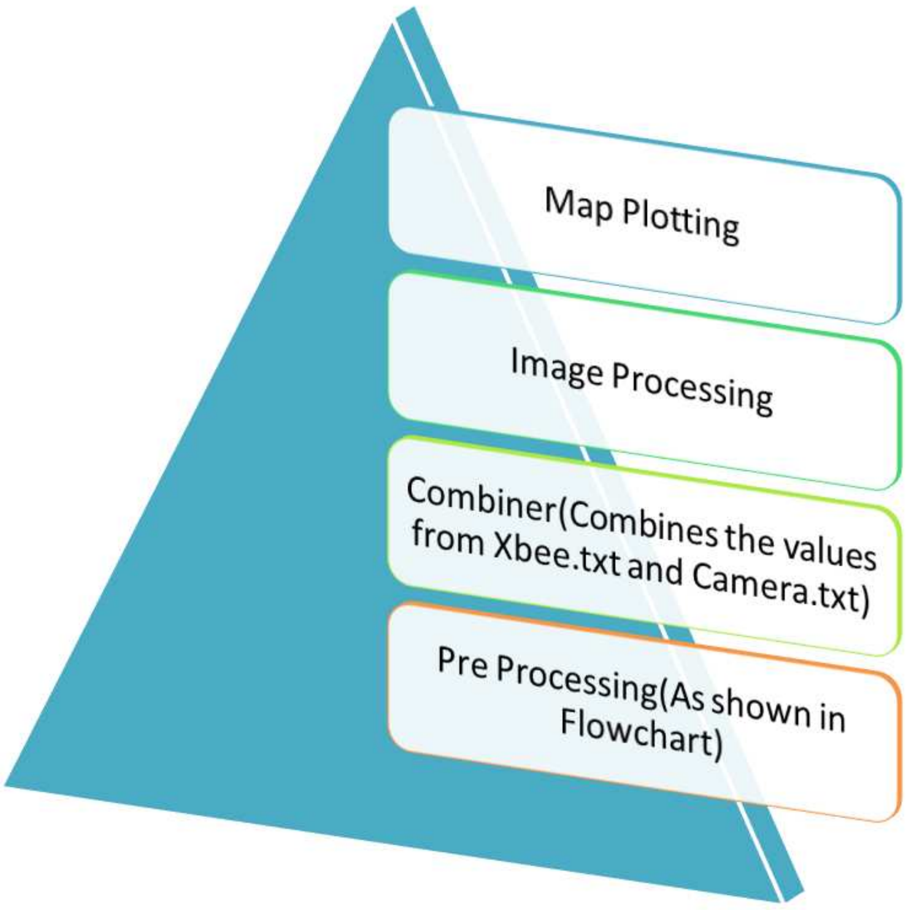

Map Plotting
Image Processing
Combiner(Combines the values from Xbee.txt and Camera.txt)
Pre Processing(As shown in Flowchart)

Figure-4 Block Diagram

2.3 Hardware Requirements

1. Firebird V ATMEGA2560
    - Three white line sensors (extendable to seven white line sensors)
    - Five GP2D12, 80cm IR range sensors covering front half part of the robot (Robot comes with one IR range
    - sensor, Four sensors are optional accessories)
    - Eight analog IR proximity sensors covering robot from all sides.
    - Eight analog directional light intensity sensors
    - Two position encoders
    - Battery voltage sensing

2. Five Sharp IR sensors mounted on Firebird Bot for sensing the objects placed in various directions and to support proper navigation. For accurate distance measurement, the robot uses IR Range sensors. These sensors give

distance from the obstacle with the millimetre accuracy. These sensors use triangulation to measure the distance from any obstacle hence there reading does not get affected by the ambient light. The robot can be fitted with five such IR range sensor’s

.

3. Five Proximity sensors mounted on Firebird for sensing the objects placed in various directions and to support proper navigation. Xbee for serial communication of sensor values. Fire Bird V robot has dedicated socket for wireless 2.4GHz ZigBee communication module. ZigBee wireless module is directly interfaced with the robot’s serial port. This module can be used by many robots to communicate with each other and with PC at the data rates up to 115Kbps. In order to communicate with a PC over wireless channel, you can use the ZigBee USB module. ZigBee wireless modules are available with 100 meters or 1000 meters communication range.
4. USB Camera mounted on firebird for localization.
5. Two PC’s with AVR Studio installed on one PC and Matlab installed on both PC’s.

2.4 Software Requirements

1. AVR Studio 4 for navigation module
2. Matlab for Image Processing.

Implementation

3.1 Code Description

Code Files :

File name	Purpose	Executes on
adc.c	Main Program, Controls Robot Navigation.	Robot
lcd.c	LCD Interfacing	Robot
Video.m	Real-Time ball tacking through image processing module and write the reading to Camera.txt	PC-1
Xbee.m	Records the reading through COM-Xbee Port and writes the reading to Xbee.txt	PC-2
Combine.m	Takes Camera.txt and Xbee.txt as input and synchronizes the timestamp and writes the value to Combine.txt	PC-1/PC-2
Map.m	Takes Combine.txt and plots 2-D map	PC-1/PC-2

3.2 Deliverable

File name	Contains	File List
C-code.tar.gz	SourceCode of programs to be burnt on Robot. Contains documentation of the code as well.	adc.c lcd.c
PC-interface.tar.gz	Contains Matlab files.	Video.m Xbee.m Combine.m

		Map.m
Documents.tar.gz	Contains Project related doc files.	Readme.txt Document.pdf Presentation.pdf

3.3 Execution Instructions

1. Compile the code adc.c in AVR Studio and burn it on firebird. Note that lcd.c should be present in the same folder as adc.c while compilation.
2. Mount the USB camera on firebird and keep it connected to pc-2. Attach Xbee transmitter to pc-1. Keep the file Video.m on pc-2 and Xbee.m on pc-1. Keep the localization shelter above the arena(around 8 ft) so that the system can calculate it's position while navigation.
3. Place firebird in the arena and switch it on. The bot will start to navigate in arena. Run the programs Video.m and Xbee.m on respective machines. Keep caution that the camera remains still on the bot till navigation is complete.
4. After the bot has navigated the arena sufficiently, stop the bot. Also stop the programs Video.m and Xbee.m.
5. Two files Camera.txt and Xbee.txt will be formed on respective machines.
6. Get the two text files on one pc and run Combine.m with text files in same folder. A new file Combine.txt will be created.
7. Run Map.m with file Combine.txt in the same folder and the map of the arena will be drawn.

3.4 Coding Guidelines

Please find attached a zip file with this document. Please refer to this code to write your own code. Important parts to be taken into consideration.

1. Use the standard "Copyright statement in your code declaring the code to be open source and property of ERTS Lab".
2. Use the standard header file "winavr_firbird.h" which will contain all the implementations of core actuation and sensing.
3. Note the commenting style. That helps Doxygen to generate documentation.
4. We have prepared the doxygen Config file in the code as well. You can simply copy paste it in the doxygen wizard. Edit the filepaths accordingly.

Innovation And Challenges

- **Proximity sensors don't transmit distances, we plotted graph for its reading to get the distance.**

  The formula for proximity sensors was not mentioned in the Manual. We tried to find out the formula on the web later on we found that proximity sensors are really very cheap and are not at all reliable. The best we made could make out of it was to not to rely on its voltage but to use it as threshold values. It was found out by rigorous testing and plotting graphs between voltage levels and actual distances.

- **For localisation, we used three different coloured (RGB) balls.**

  As mentioned above the bot has really very bad efficiency during SLAM (Simultaneous Localization and Mapping) and it frequently loses its position and it is very hard to figure out the current position because the error is multiplicative in our project. Going for a higher end solution (like GPS) was not really good option because of two reasons firstly, it is not economically good and secondly, GPS is also not accurate to centimetre accuracy. Finally we come up with innovative idea of creating local GPS that is not only economically good but also gives great efficiency.

- **Xbee data transmission breaks the order of sending. We applied delay in transmission after sending each sensor value.**

  Theoretically Xbee is supposed to transmit with a speed of 9600bps we later found out if we are sending serial data without providing delay in between can cause disordering of the characters and hence it made it worse for us to figure out the sensor values that has been send by bot through Xbee .We figured out that it can only send two character at a time preserving their sequence. So we provided a delay of 10ms after sending each pair of values. 10 ms is the optimal in our case one should figure out the value by brute force. The delay should not be very large because it will also affect the performance.

- **To synchronise the data sent by Xbee and image frames captured by camera, we used text files to store the values and then apply synchronisation algorithm.**

  When we were through with our data collection from bot (Xbee) and camera(Angle of rotation and coordinates of bot) we found that there is a huge delay of 4 seconds between Xbee reading and processed image frame for that time instance. We figured out that the hunch was in plotting map in real-time on a fly. We conclude that this is not a good idea. So we opted for a distributed system solution in which Xbee reading were recorded in a text file on one System and the Readings from the camera on other system. We also synchronized the time of both the systems up to accuracy of fraction of second. We also developed a separate algorithm for merging the synchronized data files.

- **One big problem is the accuracy of sensors. Also they are colour dependent. (best with white colour)**

  This was one problem that is theoretically not possible but we found out that the accuracy of sensor values is dependent on colour of obstacles. So we opted for white coloured arena and obstacles.

Testing

5.1 Test Cases

S.No	Test Case	Actual Output	Expected Output	Correction
1.	Localize the Bot using local GPS	Get the actual position by simple image processing on captured image	Estimating Actual location of the bot on the field	Not needed
2.	Reliable sensors values communication using Zig-Bee	Not good enough because it is not a reliable serial communication path	Reliable communication path	Reliable serial wireless communication path needed
3.	Object Exploration using sensors	Not good enough due to poor quality sensors	Reliable sensors values	Reliability can be assured by using good quality sensors
4.	Conversion of view to world coordinate system	Devices work in 3rd coordinate system and give approximate point	Accurate position of bot in world coordinate system	Finding out the coefficient to calibrate the result

5.2 Snapshots

Test Case-1(Rectangular field without object)

Figure-5 Field without object

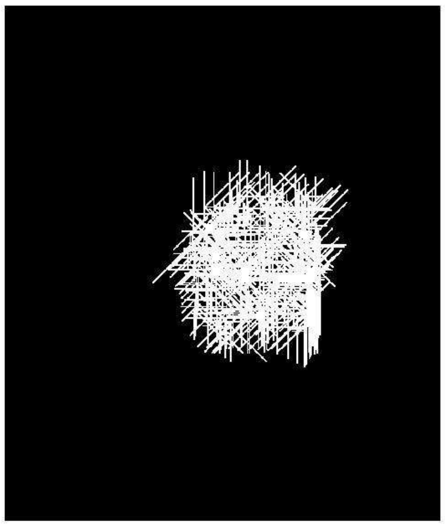


Figure-6 Map traced using sensors values

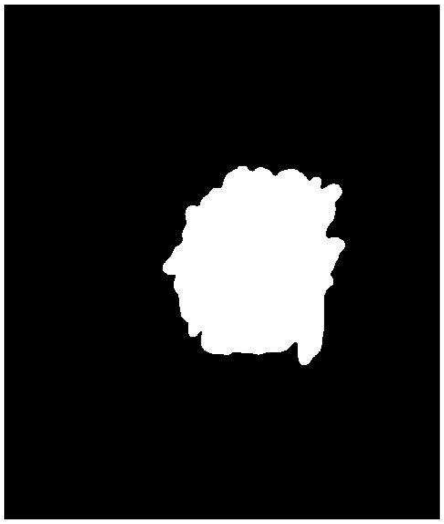


Figure-7 Map traced after image processing

Test Case-2 (Rectangular field with object)

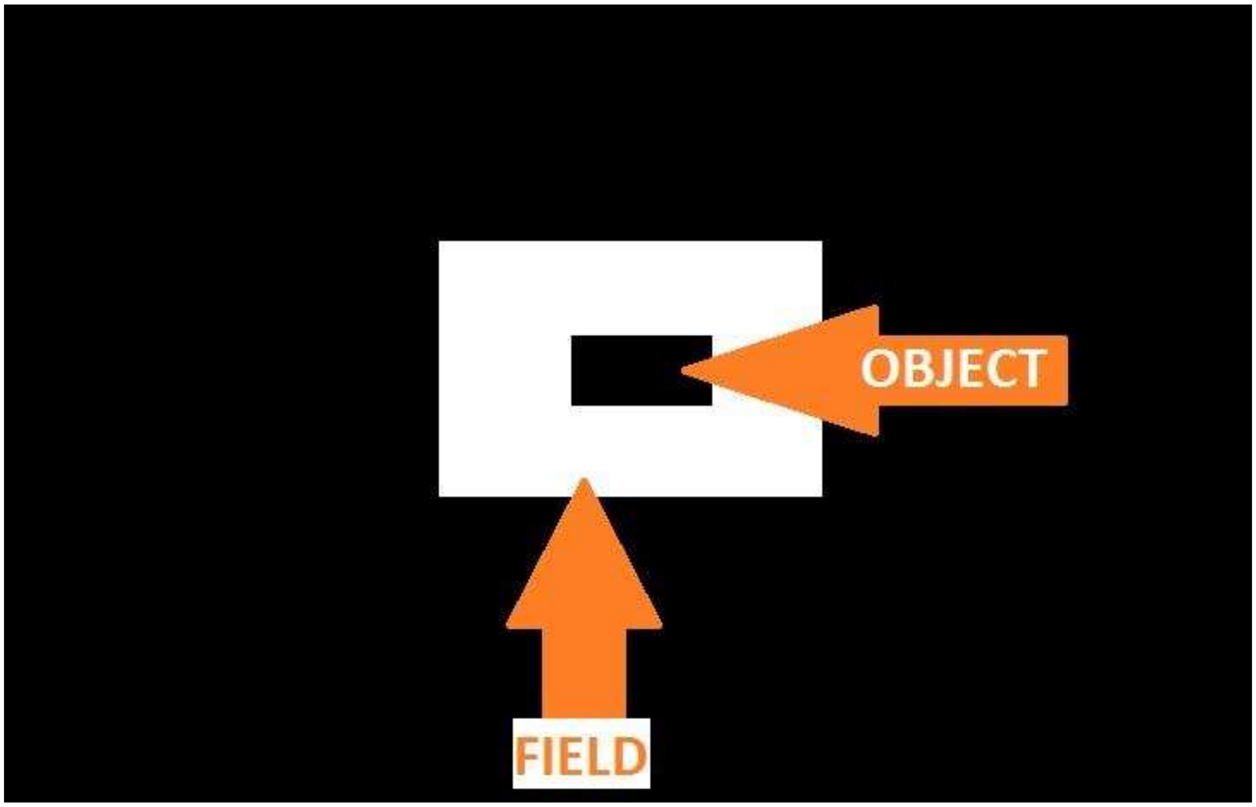


Figure-8 Field with object

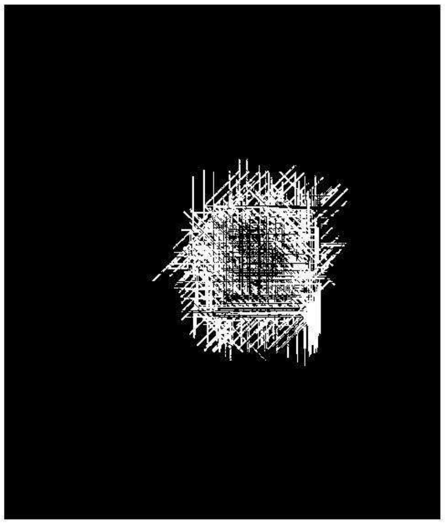


Figure-9 Map traced using sensors values

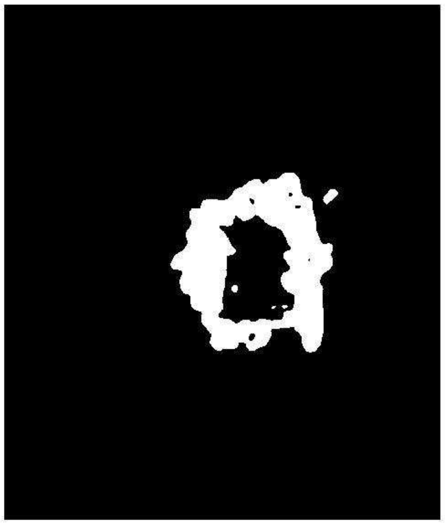


Figure-10 Map traced after image processing

Conclusion

6.1 What we set out to do:

- We aimed of building an autonomous bot which will navigate an unknown territory.
- It will send the video to the terminal and draws the map of arena.
- Through face detection it will send us the images of persons from the unknown location.
- We thought of locking a face and then making the bot follow that face silently sending the location of that target.

6.2 What was implemented:

- Unknown environment exploration and map tracing has been implemented and experimented in the test plans.
- The Bot is able create a topological map of the unknown environment.
- The algorithm has been tested against several different configurations of fields.
- The face detection and locking in real-time was not feasible with the limited processing power of bot and camera.

6.2 Future Scope

1. Global GPS implementation to reliable localization of Bot.
2. Number of the Bots on exploration in different type of environments to provide efficiency and reliability in this method.
3. Better quality sensors for accurate object exploration.

References

1. Fire Bird V ATMEGA2560 Hardware Manual

2. Fire Bird V ATMEGA2560 Software Manual

3. http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

4. http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf

5. http://www.openslam.org/

6. http://www.ai.rug.nl/~gert/as/

7. http://www.mathworks.com/help/techdoc/index.html

README : USER GUIDE

Project Title: MAP TRACER
Team Members: Group 13

Name	Roll No.	Email Id
Arpit Malani	10305901	malani@cse.iitb.ac.in
Hermesh Gupta	10305080	hermesh@cse.iitb.ac.in
Rahul Nihalani	10305003	rahuln@cse.iitb.ac.in
Vivek V Velankar	10305050	vivekvelankar@cse.iitb.ac.in

Project Objective

Exploration of an unknown environment searching operations inside buildings, caves, tunnels and mines are sometimes extremely dangerous activities. The use of autonomous bots to perform such tasks in complex environments will reduce the risk of these missions.

Hardware Platform

1. Five Proximity sensors mounted on Firebird for sensing the objects placed in various directions and to support proper navigation.
2. Firebird V ATMEGA2560
3. Five Sharp IR sensors mounted on Firebird Bot for sensing the objects placed in various directions and to support proper navigation.
4. Xbee for serial communication of sensor values.
5. USB Camera mounted on firebird for localization.
6. Two PC's with AVR Studio installed on one PC and Matlab installed on both PC's.

Software Requirements

1. AVR Studio 4 for navigation module
2. Matlab for Image Processing.

Instructions For Included Files

1. **PC-Interface.tar.gz-** This archive includes the matlab source code files which are-
   a. Xbee.m – It records the sensor values in a file called Xbee.txt.
   b. Video.m- It records the camera image frame data in a file called camera.txt.
   c. Combine.m- It combines above mentioned files to a single data file called combine.txt.
   d. Map.m- Using the file generated above(combine.txt) it draws the map.

2. **C-Code archive.tar.gz-** This archive includes c source code which executes on BOT. Files are-
   a. adc.c- It is responsible for the navigation of the bot. It senses the sensor values, converts them to distances and sends to Xbee sender.
   b. lcd.c- It displays the actual free distances in 5 direction on the lcd screen.

3. **Documents.tar.gz-** It includes all the documentation and presentation files. It also includes the doxygen files.
4. **README-** It includes the execution instructions.

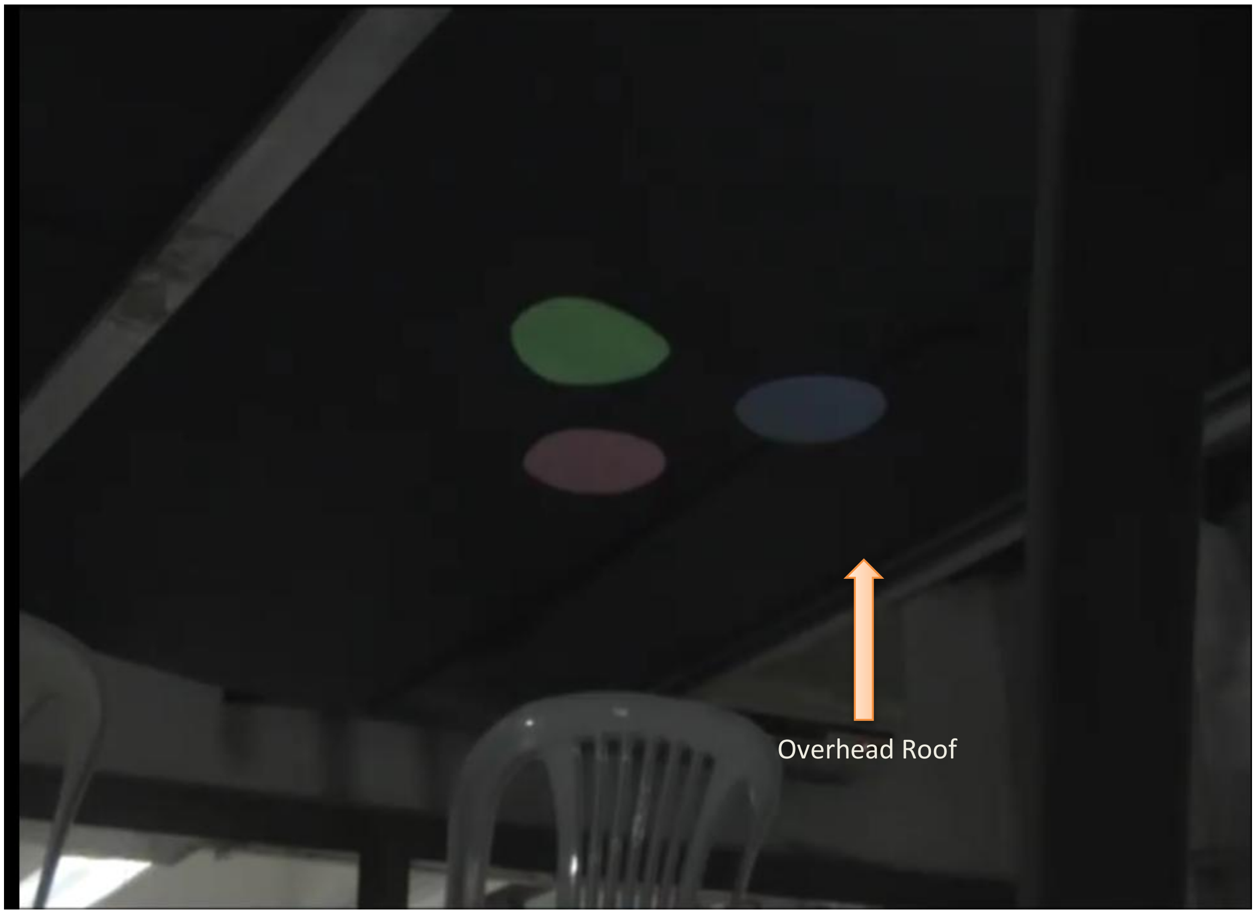
Execution Instructions

1. Compile the code adc.c in AVR Studio and burn it on firebird. Note that lcd.c should be present in the same folder as adc.c while compilation.
2. Mount the USB camera on firebird and keep it connected to pc-2. Attach Xbee transmitter to pc-1. Keep the file Video.m on pc-2 and Xbee.m on pc-1. Keep the localization shelter above the arena(around 8 ft) so that the system can calculate it's position while navigation.
3. Place firebird in the arena and switch it on. The bot will start to navigate in arena. Run the programs Video.m and Xbee.m on respective machines. Keep caution that the camera remains still on the bot till navigation is complete.
4. After the bot has navigated the arena sufficiently, stop the bot. Also stop the programs Video.m and Xbee.m.

5. Two files Camera.txt and Xbee.txt will be formed on respective machines.
6. Get the two text files on one pc and run Combine.m with text files in same folder. A new file Combine.txt will be created.
7. Run Map.m with file Combine.txt in the same folder and the map of the arena will be drawn.

Project Setup:-

1. This picture shows the setup for localised GPS, in which three balls are used to track the location of bot in the arena.



2. This picture shows the movement of bot in the arena along with the view of overhead camera mounted on top of bot.