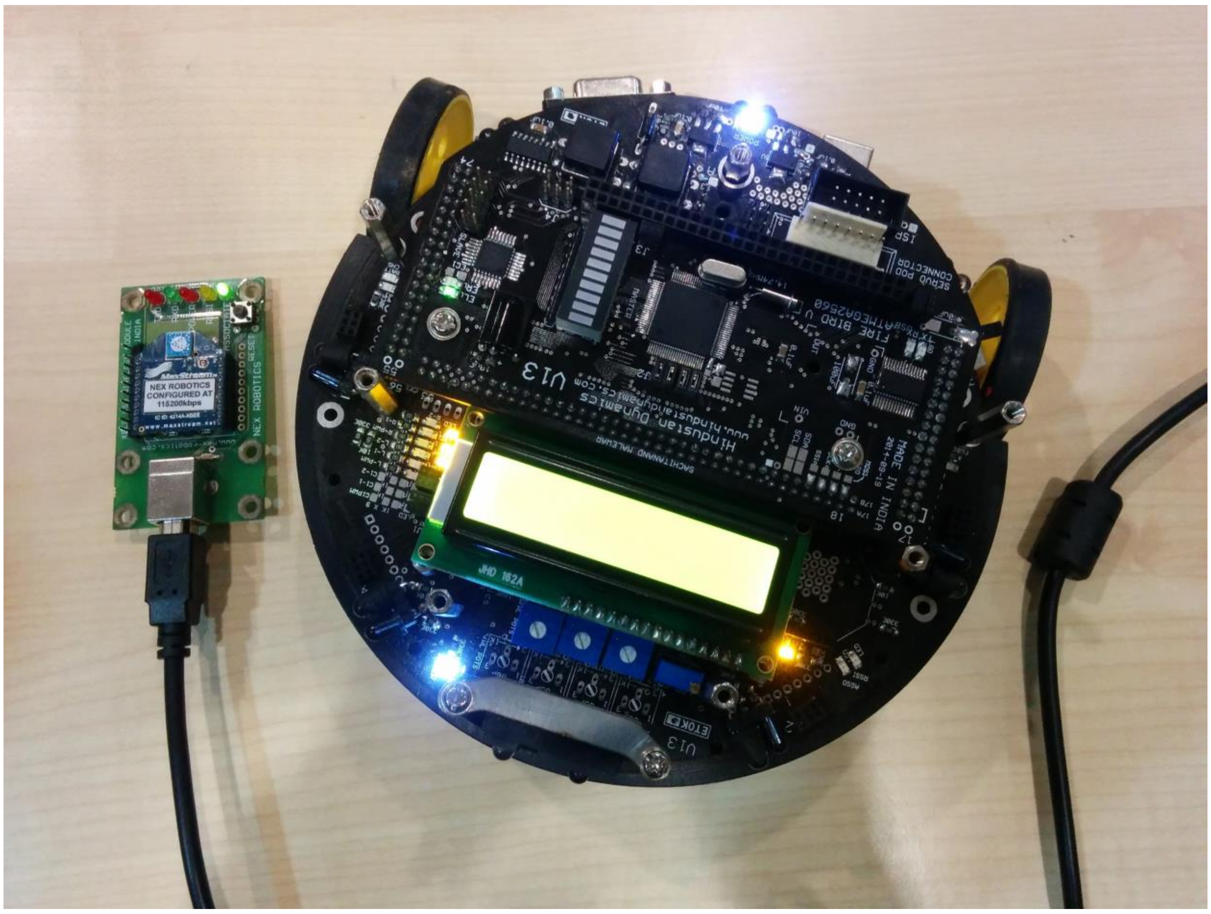
AUTONOMOUS OBSTACLE DETECTING PATH TRACER

PROJECT REPORT

CS 101 EMBEDDED SYSTEMS PROJECT



TEAM ID – 495

MEET UDESHI (TL)	14D070007
SIDDHANT GARG	14D070027
PRAKIRT RAJ JHUNJHUNWALA	140070027
NISHANT DHAMA	14D070064

CONTENTS

1. Introduction

Nowadays, automation is dominating the field of Navigation and mapping. With the advent of robotics and artificial intelligence this job has become easier and vast as we are able to reveal the places beyond man's reach along with mapping their topology.

This idea is further being implemented in the concept of Driverless cars.

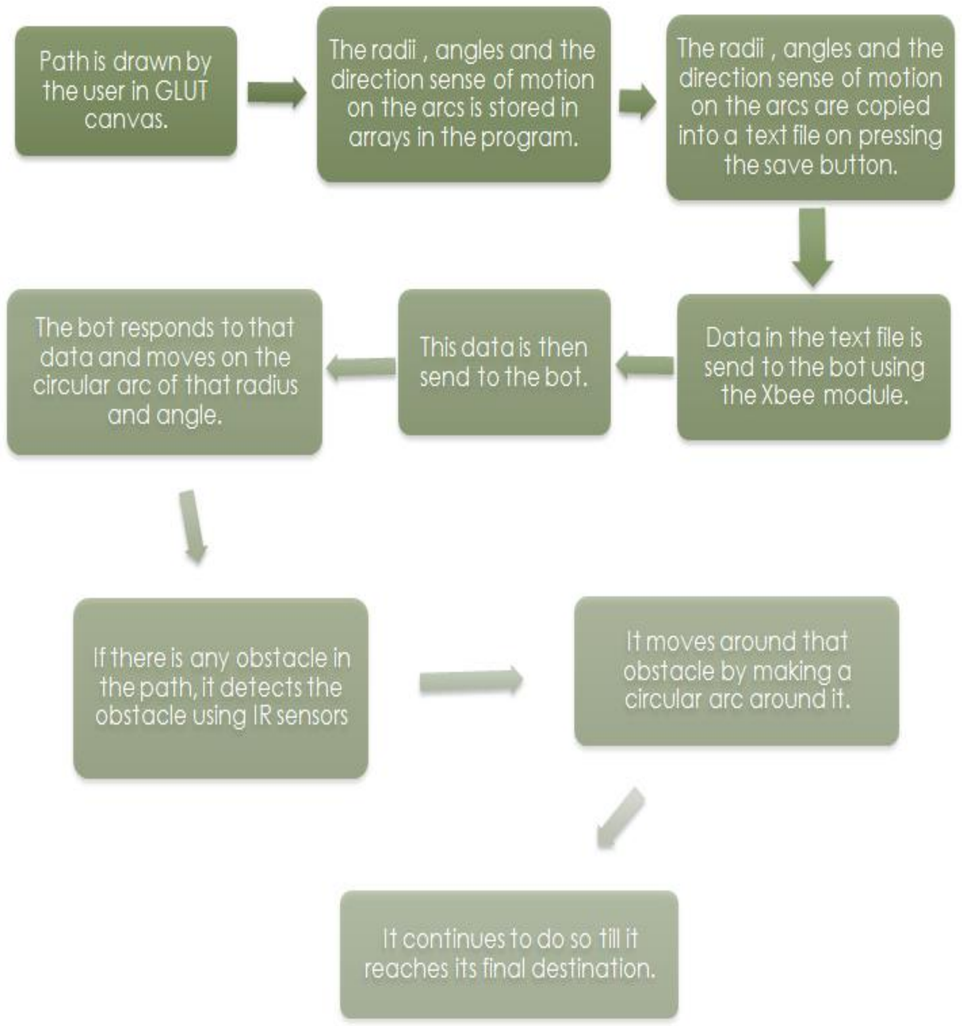
Inspired by these developments we decided to create an "Autonomous Obstacle Detecting Path Tracer" using the Firebird V.

The project is aimed at implementing a system which can follow a path in an arena whose shape is drawn by the user on the laptop. This path would be given as input to the bot. The bot will travel on this path avoiding any obstacle which falls in the path by generating a new path around the obstacle. Thus suiting its name- 'Autonomous Obstacle Detecting Path Tracer'.
Additionally, we have been able to implement the path generation in an Android device rather than in a Laptop.

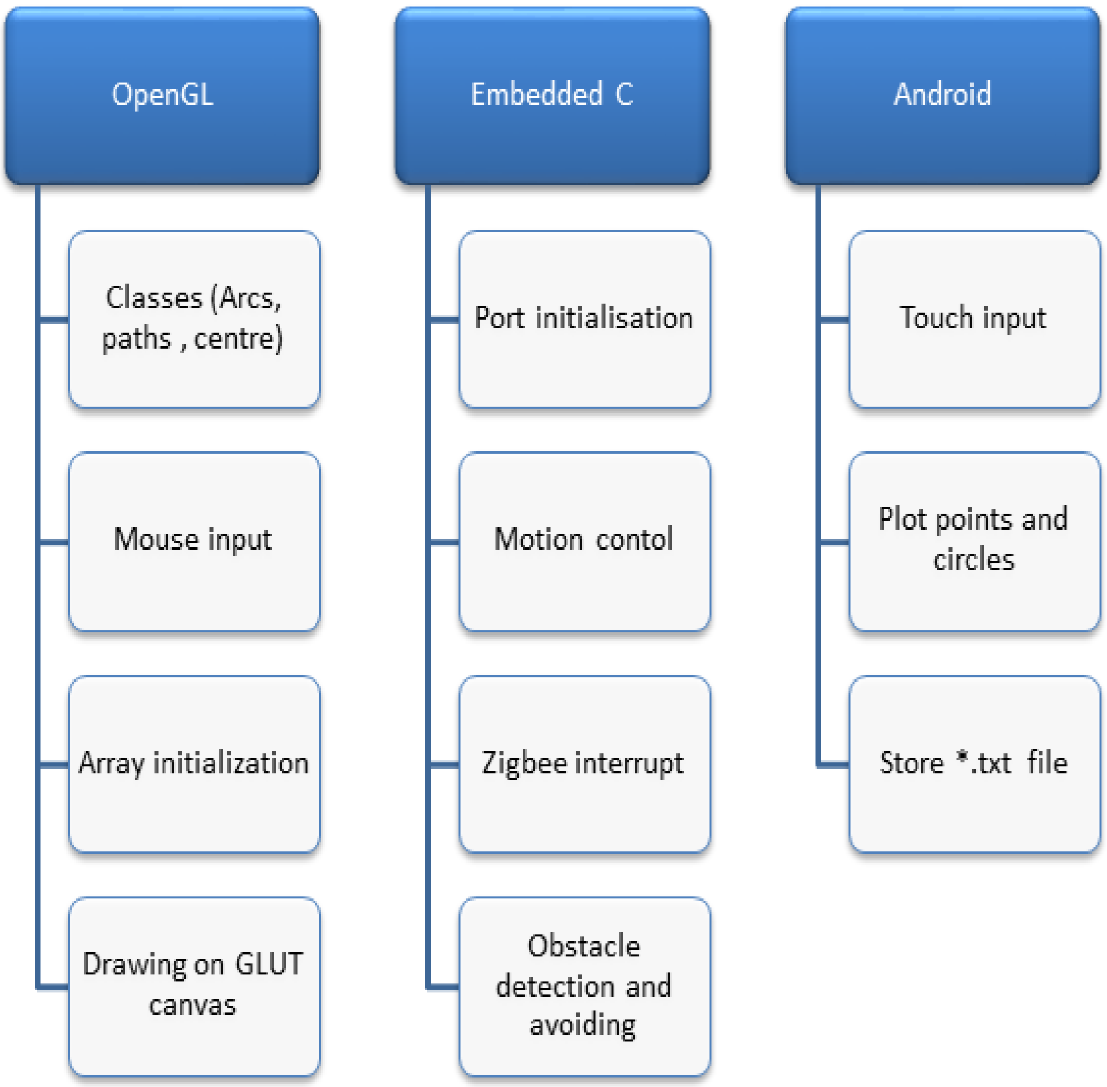
2. Why this Project?

On getting the firebird v bot, there were just so many things that we could have done. We thought of implementing something by which we could learn to program and use the firebird ATmega v as well as implement the things that we have learnt in CS 101. We could incorporate the code for optimized path input from the user (which would require things learnt in the class) along with coding the ATmega bot for following the path. We thought that this project has great implementation and future scope and thus decided to work on this topic.

FLOWCHART SHOWING EXECUTION OF THE PROJECT


Path is drawn by the user in GLUT canvas.
The radii , angles and the direction sense of motion on the arcs is stored in arrays in the program.
The radii , angles and the direction sense of motion on the arcs are copied into a text file on pressing the save button.
Data in the text file is send to the bot using the Xbee module.
This data is then send to the bot.
The bot responds to that data and moves on the circular arc of that radius and angle.
If there is any obstacle in the path, it detects the obstacle using IR sensors
It moves around that obstacle by making a circular arc around it.
It continues to do so till it reaches its final destination.

SOFTWARE ARCHITECTURE



3. Constraints

- Path given by the user should be continuous and differentiable (smooth).
- End points of the path should lie inside arena.
- Arena should be flat and traversable. We have not included negative obstacle detection- manholes, pits, etc.
- The Position encoding of the Bot is not very accurate, so the results are constrained by this limitation.

4. Assumptions and Dependencies

- At least one possible path (without having any obstacles in the path) should be present from the starting position to the final position considering the dimensions of the bot.
- The obstacle should be such that its base is smaller than 15 cm x 15 cm. This is the value used by us for test cases, but this can easily be changed according to the user's requirements.
- The wheels of the Bot do not slip on the ground as this will bring errors in position encoding which will give wrong results.

5. Modules

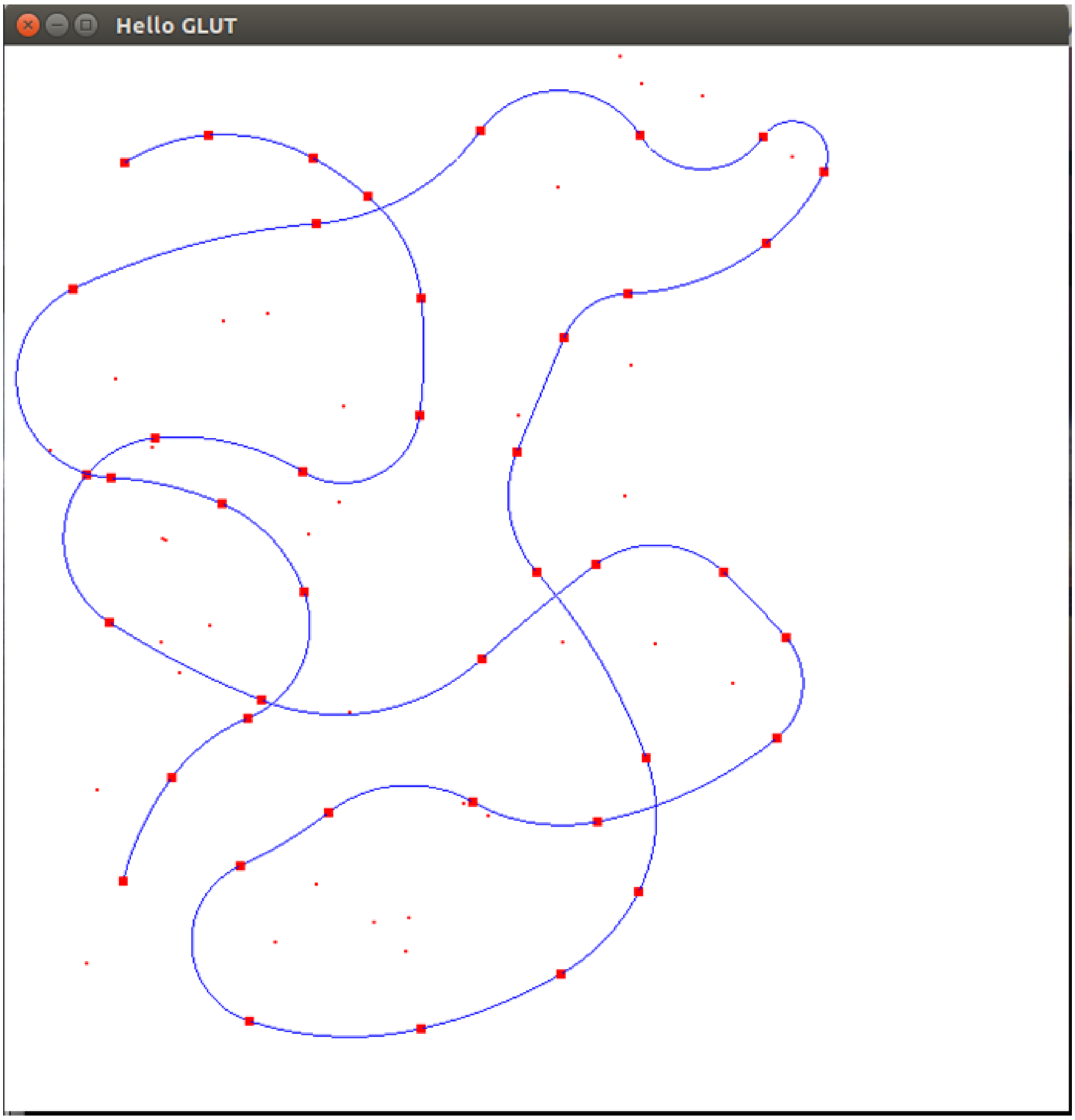
5.1 Taking input from the user and path generation

We first encountered the problem of how to break the path input from the user and feed it to the bot.
Now an easy approach would have been to break the path into points and join the points using straight lines. However, on changing the path from one line to another, the bot would have to rotate about its center of mass and align itself to the new direction and this would take time.

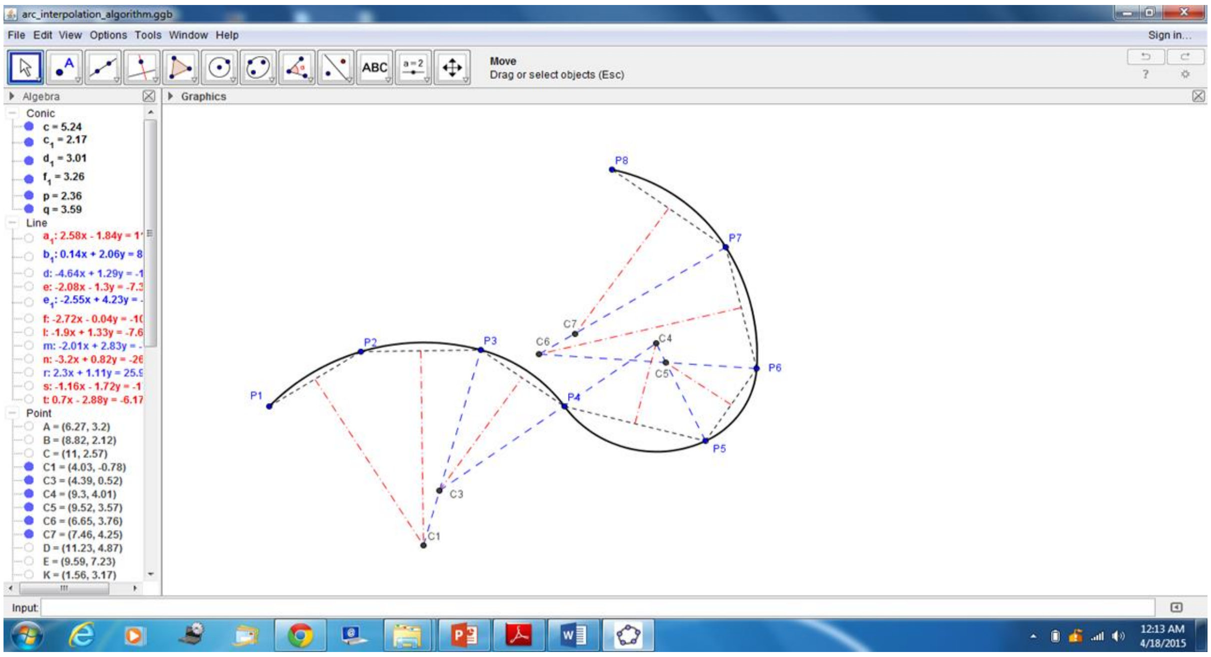
As a solution to this, we came up with the idea of implementing the path as a series of circular arcs wherein each arc is tangent to the previous arc at the intersection point . Using this algorithm , the bot will not have to change its direction and can directly continue on the new path.

Here we found out the coordinates of the center of the (n+1)th arc by having the coordinates of end point and centre of the nth arc and the end point of the (n+1)th arc. Since both the arcs are tangent to each other, the centres of both the arcs and the start point of the (n+1)th arc will lie on the same straight line. Also, the centre of the (n+1)th arc will lie on the perpendicular bisector of the line joining the start and end points of the (n+1)th arc. Thus by solving for the intersection points of the above two lines specified, we get the centre of the (n+1)th arc. This will also provide us with the angle of the arc and the direction of sense of motion on the arc- clockwise or anticlockwise.

This is a screenshot of how the input window looks like. The path is created in series of circular arcs. Each arc is tangential to the previous arc at the point common to both the arcs.

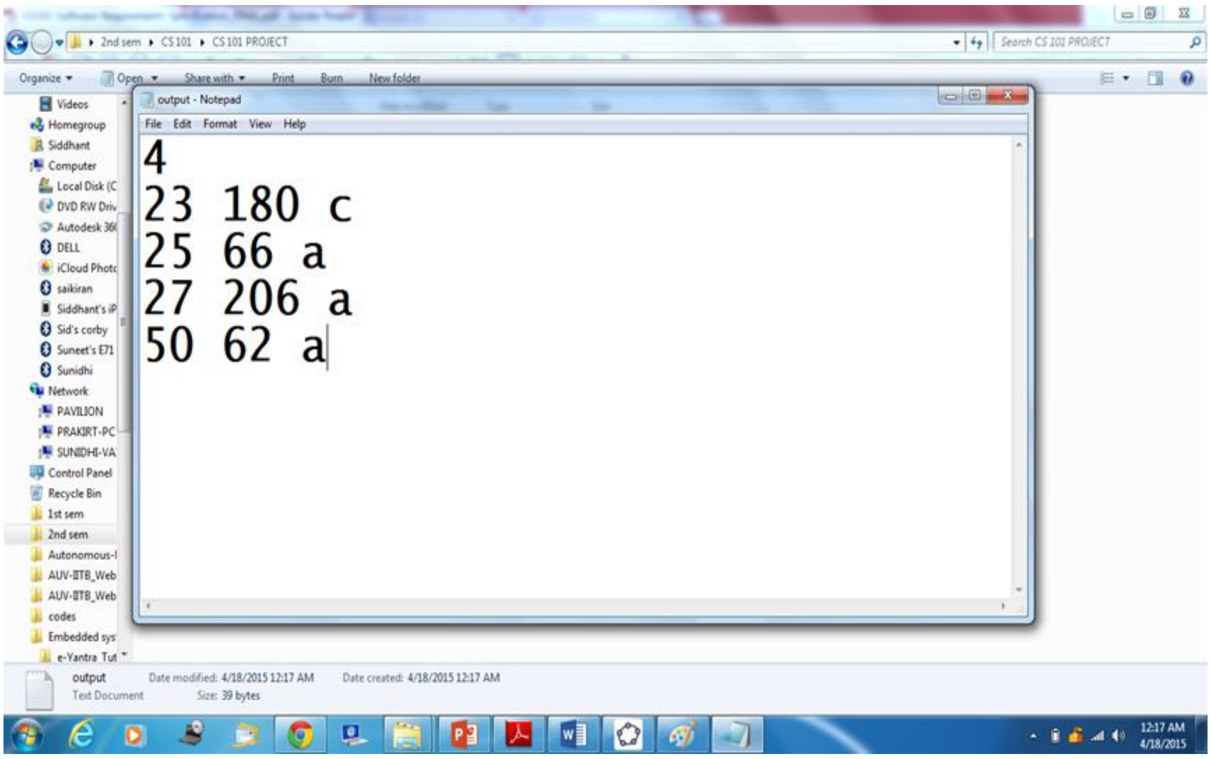


Representation of the mathematical part of our path generation on geogebra software.


arc_interpolation_algorithm.ggb
File Edit View Options Tools Window Help
Move
Drag or select objects (Esc)
Algebra
Graphics
P1
P2
P3
P4
P5
P6
P7
P8
C1
C3
C4
C5
C6
C7
Input:
12:13 AM
4/18/2015

```
Conic
c = 5.24
c_1 = 2.17
d_1 = 3.01
f_1 = 3.26
p = 2.36
q = 3.59
Line
a_1: 2.58x - 1.84y = 1
b_1: 0.14x + 2.06y = 8
d: -4.64x + 1.29y = -1
e: -2.08x - 1.3y = -7.3
e_1: -2.55x + 4.23y = -
f: -2.72x - 0.04y = -10
l: -1.9x + 1.33y = -7.6
m: -2.01x + 2.83y = -
n: -3.2x + 0.82y = -26
r: 2.3x + 1.11y = 25.9
s: -1.16x - 1.72y = -1
t: 0.7x - 2.88y = -6.17
Point
A = (6.27, 3.2)
B = (8.82, 2.12)
C = (11, 2.57)
C1 = (4.03, -0.78)
C3 = (4.39, 0.52)
C4 = (9.3, 4.01)
C5 = (9.52, 3.57)
C6 = (6.65, 3.76)
C7 = (7.46, 4.25)
D = (11.23, 4.87)
E = (9.59, 7.23)
K = (1.56, 3.17)
```

The text file that was created from the path generation code on glut canvas.



```
4
23 180 c
25 66 a
27 206 a
50 62 a
```

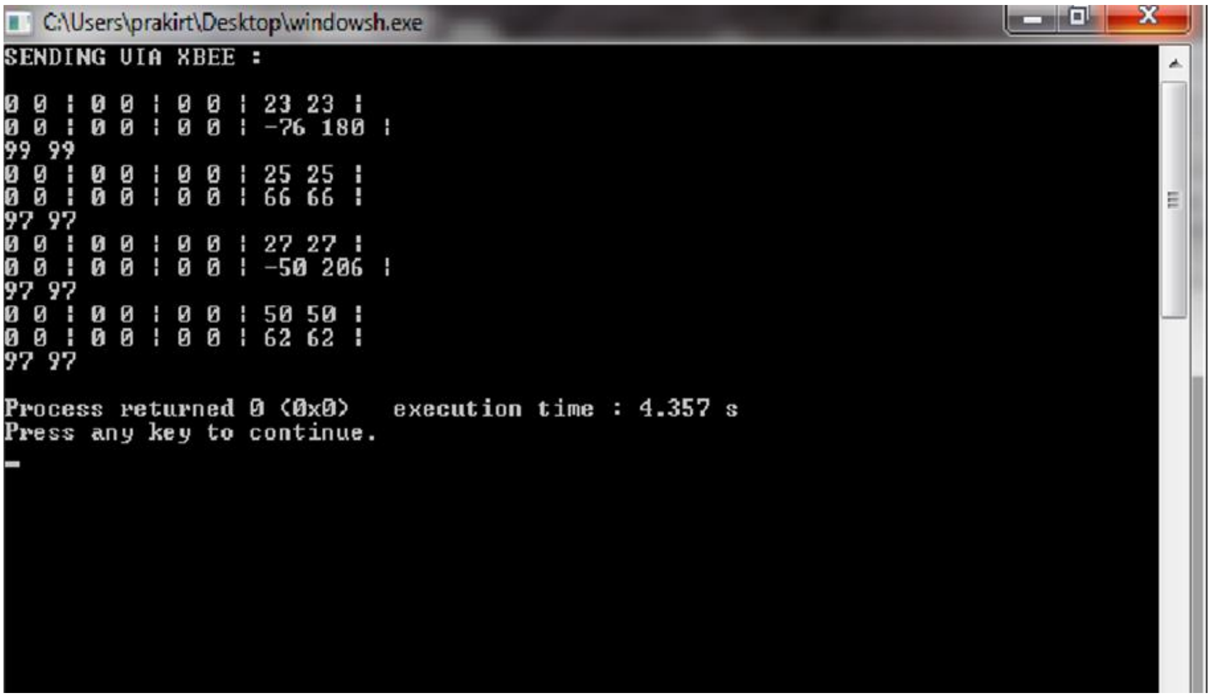
5.2 Motion of the Bot on the path and Obstacle Detection

Since the PWM (Pulse Width Modulation) of the Firebird V was not accurate enough for our algorithm, we executed the path following by position encoding. Each arc was broken into a fixed number of segments (say 5) and then the bot will move along the chords joining the points rather than the arc joining them. This ensured a better accuracy of path following according to the input given on the canvas by the user.

Also in case the bot encounters an obstacle during the path, the bot will stop at a certain distance from the bot (a threshold value defined by us - 15cm) .It will first rotate about its centre of mass to align itself tangentially to circle of radius 15 cm around the obstacle. Then it will move on this circle until it reaches the point where the circle cuts the arc again. Here again it will rotate to realign itself to the path direction. Now it will continue to complete the path. This algorithm will also work in case of extended obstacles as this algorithm will be executed repeatedly in recursion.

5.3 Wireless communication between the laptop and Firebird V via Xbee module

The radius, angle of the arc and the direction of motion on the arc-clockwise or anticlockwise was given to the bot via the Xbee module. This information is stored in a .txt file format from the GLUT canvas and a c file was created using a windows.h header file which broke in the information and communicated to the bot byte by byte.



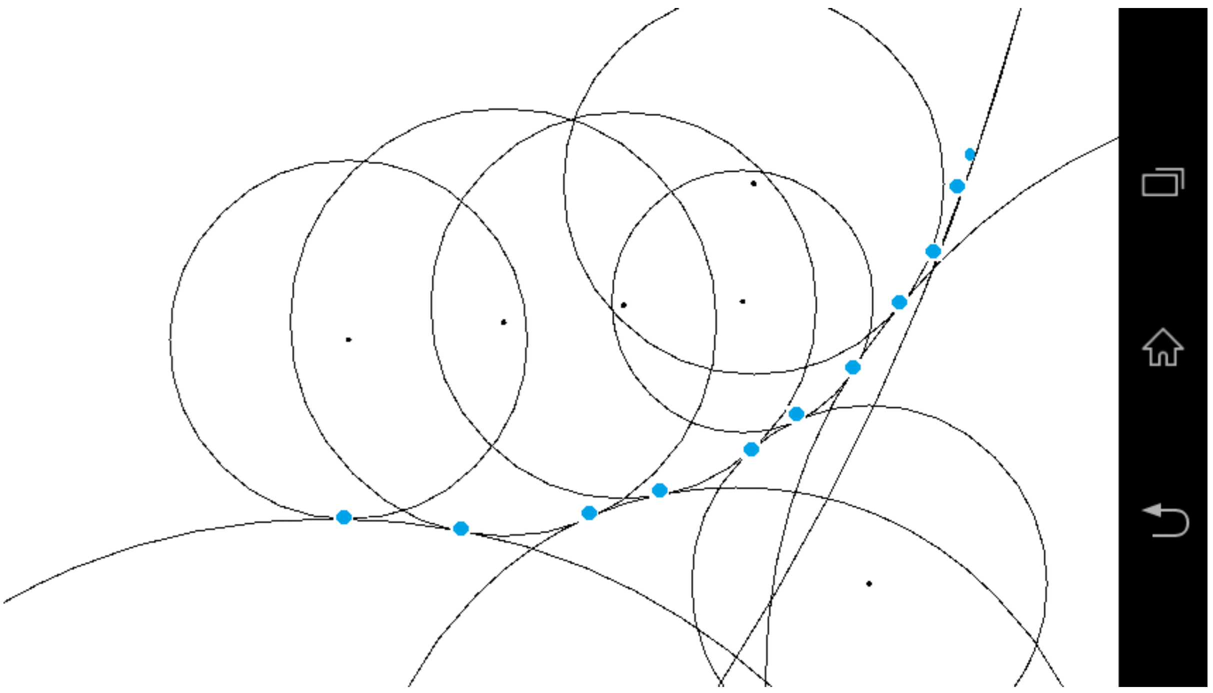
```
SENDING VIA XBEE :

0 0 | 0 0 | 0 0 | 23 23 |
0 0 | 0 0 | 0 0 | -76 180 |
99 99
0 0 | 0 0 | 0 0 | 25 25 |
0 0 | 0 0 | 0 0 | 66 66 |
97 97
0 0 | 0 0 | 0 0 | 27 27 |
0 0 | 0 0 | 0 0 | -50 206 |
97 97
0 0 | 0 0 | 0 0 | 50 50 |
0 0 | 0 0 | 0 0 | 62 62 |
97 97

Process returned 0 (0x0)   execution time : 4.357 s
Press any key to continue.
_
```

5.4 Path generation in Android Phone

The execution of the path that was initially done in the GLUT canvas was also achieved in an Android device. This module was an additional achievement that we managed to do with the paucity of time. However, the android graphics libraries did not have an arc function, so we left the path generation till formation of circles. In the below image, the path is formed through the blue dots.



6. Challenges faced and how they were overcome

SERIAL NUMBER	CHALLENGES FACED	HOW WE OVERCAME THEM AMELIORATION
1.	Thinking up of some algorithm by which the bot can follow the path which the user wants it to follow as close as possible.	We came up with idea of breaking the whole path into a series of circular arcs wherein each arc is tangential to the previous arc. This would reduce the amount of time spent by the bot on taking turns about its centre of mass at the same location to align itself with the next line segment.
2.	Wireless communication from the laptop to the bot by using libXbee was not functional.	We decided to include windows.h header file to directly read/write to the Xbee COM port in our code for wireless communication.
3.	The execution of the arcs on the ground by the bot was not appropriate. We needed the bot to move in circular arcs of different radii but this needed extremely accurate determination of velocities of the two wheels of the bot. However the PWM (Pulse Width Modulation) based velocity control of the bot is extremely inaccurate as found out by us in several of our test data cases (It was so inaccurate that the bot was running on arcs of the same radius for inputs of 50 cm and 100 cm separately).	To overcome this major problem that we had, we came up with the idea of having to give up the idea of using PWM. We decided to break the circular arc into a particular number of equal segments and then instructing the bot to move on the chords of these segments rather than the arc. Though this may sound as if contradicting our initial purpose for writing the codes for path generation in terms of arcs, it worked quite appropriately in our test cases. The bot will surely have to spend some time in changing its direction , but the angle by which it will have to change its direction will be small enough at each interval. For future application, if a better PWM based velocity control can be achieved, our initial algorithm will be very efficient.
4.	We made the code for path generation using GLUT (Open GL) in Ubuntu. However windows.h header file for ZigBee would work only on windows. Our path was getting generated on Ubuntu but	We tried to download the libraries for Open GL on windows and try to compile the code. But due to time constraint, we thought of sending the data about the path generated in Ubuntu via a .txt file to a laptop having Windows and then using that for path following.

	our other code for communication for the Bot was on Windows.	
5.	During obstacle detection, Inaccurate sensor value due to external lighting and obstacles of different color was a big problem. Also the additional sharp sensors that we thought of using were not properly functional.	Windows and other sunlight sources were covered by black cloth to reduce IR lights. An appropriate Threshold sensor value was set based on the size of the obstacle we were using.
6.	Position encoding of the Bot is not accurate. Inaccurate sensors and slipping of wheels on smooth surface.	This is a limitation of the Bot and we have not been able to overcome this. As this problem could be solved only by using a bot with better sensors.

7. Future Scope, Applications and Innovations

1.] A major future application of our project is the

‘Driverless Car’.

How it can be achieved?

This project also needs the car to follow the path given by the user and at the same time detect obstacles in its path- humans, other cars and vehicles,etc. So if we implement our project on a larger scale with very accurate PWM, position encoding, better large scale sensors, etc. We can formulate a driverless car. Actual cars with rack and pinion mechanism need a continuous and differentiable path to move which is being provided by our algorithm.

Application?

This will have great application for the physically handicapped people and the blind people who then will be able to independently run a vehicle.

2.] A major future application of our project is the

'Walking Aid for Blind People'.

How it can be achieved?

By attaching a stick or a rope with the bot along with additional sensors on the stick (rope) to detect any obstacles in the blind person's pre specified path, he/she can be alerted by audio to change his/her path.

Application?

This will have great application for the blind people who then will be able to independently walk and move about completely safe without being dependent on others for help.

8. References

1. Firebird V Hardware and Software Manuals.
2. https://www.opengl.org/resources/libraries/glut/ as Graphic Library reference
3. ZigBee Reference Manual.
4. https://www.youtube.com/watch?v=Boy5asltEJw to configure Xbee using X-CTU.
5. https://msdn.microsoft.com/en-gb/default.aspx to check Windows.h functions.

You can see the project video on https://youtu.be/dD4qjS22m34 and the screencast and spoken tutorial video on http://youtu.be/86wnkCGIsLE .The project is uploaded on https://github.com/udiboy1209/Autonomous-Path-Follower repository.