Software Requirements Specification

CS101 Project 2015

Autonomous Obstacle Detecting Path Tracer

(Group CUSE)

Meet Udeshi - 14D070007
Siddhant Garg - 14D070027
Prakirt Raj Jhunjhunwala - 140070027
Nishant Dhama - 14D070064

Table of Contents

1. Introduction

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 (or an android device). 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’. If time permits , we will aim to implement the path generation in an Android device rather than in a Laptop.

1.1 SCOPE

1. This project if enhanced and improved further can professionally be used as a navigation and mapping algorithm for a driverless car. Improving the sensors can lead to this Bot finding applications in analyzing a terrain which may be difficult to traverse and thus may find applications in Space Missions, Army and military purposes, etc.

2. An altogether different application where this project might be useful is travelling purpose for blind people. The Bot can detect what obstacles lie near the person and will suitably generate a safe path to the final destination which can then be passed on to the blind person through audio.

1.2 ACRONYMS, DEFINITIONS AND ABBREVIATIONS

1. **SRS (Software Requirements Specification)** - A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.

2. **User-** The person operating the laptop (or Android device) who will draw the path to be followed by the Bot.

3. **Obstacle-** Any object or feature of the arena which may get detected by the IR sensors on the Bot and may not allow the passage of the Bot along the path which is defined (considering the dimensions of the Bot).

4. **Path-** The connection of various points from a certain initial position to a specific final position. In this case, the path is drawn by the user on laptop (or an android device) connecting any 2 points in the arena specified.

2. Overall description

Product Perspective: Path follower and automatic obstacle detection and mapping.

Product Functions: The project is aimed at implementing a system which can follow a path in an arena, avoiding obstacles in its way by generating a new path.

User Characteristics: The android device user will operate the application and send the required path via Bluetooth (ZigBee) [For academic purposes].

Constraints: Path given by the user should be continuous and smooth. End points of the path should lie inside arena. Arena should be flat and traversable

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 area where the Bot moves is traversable by the Bot.

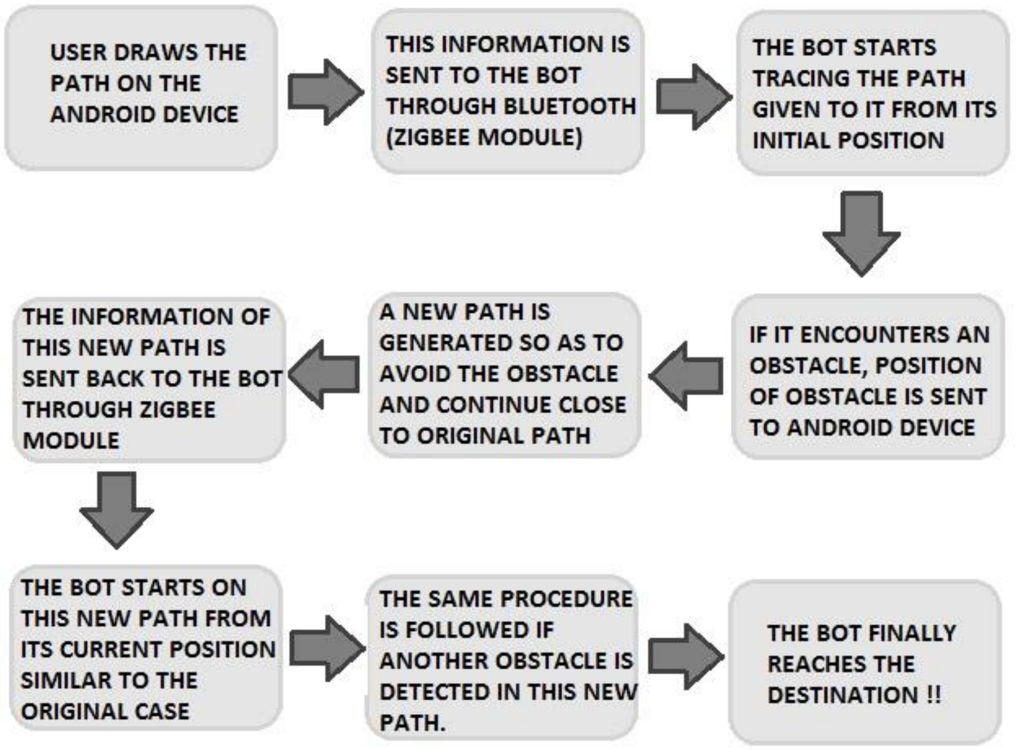
Requirements Subsets:

- Firebird V
- 2 IR sharp sensors apart from that present on Firebird V
- ZigBee modules and adapter.

3. Details

3.1 Functionality

- User will draw a path on android device. The path will be stored as a set of points.
- The set of points will be interpolated with circular arcs, such that each consecutive circular arc is tangent to each other
- The data of this interpolated curve (radius of curvature data) will be sent to the Firebird V robot via Bluetooth (ZigBee)
- This path will be traced by the Firebird V robot.
- When an obstacle is detected by the sharp sensors, position of this obstacle will be sent to android device.
- An alternate path will be generated using that position data.
- Bot will be sent updated path.


USER DRAWS THE PATH ON THE ANDROID DEVICE
THIS INFORMATION IS SENT TO THE BOT THROUGH BLUETOOTH (ZIGBEE MODULE)
THE BOT STARTS TRACING THE PATH GIVEN TO IT FROM ITS INITIAL POSITION
IF IT ENCOUNTERS AN OBSTACLE, POSITION OF OBSTACLE IS SENT TO ANDROID DEVICE
A NEW PATH IS GENERATED SO AS TO AVOID THE OBSTACLE AND CONTINUE CLOSE TO ORIGINAL PATH
THE INFORMATION OF THIS NEW PATH IS SENT BACK TO THE BOT THROUGH ZIGBEE MODULE
THE BOT STARTS ON THIS NEW PATH FROM ITS CURRENT POSITION SIMILAR TO THE ORIGINAL CASE
THE SAME PROCEDURE IS FOLLOWED IF ANOTHER OBSTACLE IS DETECTED IN THIS NEW PATH.
THE BOT FINALLY REACHES THE DESTINATION !!

3.2 Supportability

- We will ensure that our code is readable and reusable, and also work towards its portability.
- We will follow the firebird specific guidelines, file structure and the project submission format as specified in the ERTS lab coding guidelines.
- We will use standard C and ‘Embedded C’ libraries.
- We will use Doxygen for our documentation.

3.3 Design constraints

- The range of the ZigBee module will determine the range of the user to control the bot within certain radius.

3.4 On-line User Documentation and Help System requirements:

- We will use GitHub for the distributed revision control and source code management.
- We will use Doxygen for online documentation.

3.5 Interfaces

3.5.1 User Interfaces

- Draw path using touchscreen based android device
- View interpolated path
- View position of bot on path
- View obstacle position
- View updated path in different colour

3.5.2 Hardware Interfaces

Sensors:

- 3 IR sharp sensors
- 8 analog IR proximity sensors

ZigBee module:

- Connected to ATmega 2560 through RxD0/PE0 (pin 2) and TxD0/PE1 (pin 3)
- Connected to the serial port of the computer
- AVR ISP mkII for programming the microcontroller

3.5.3 Software Interfaces

- AVR GCC: Compiler for Ubuntu
- AVRDUDE programmer for Ubuntu
- AVR Studio 6.0 (On Windows8)
- AVR Bootloader
- Android 2.3.3 SDK (API 10)
- Android studio

3.5.4 Communication Interfaces

- Wireless ZigBee communication between the Laptop and Firebird V Bot to send information regarding the path to the Bot. (2 ZigBee modules and one adapter)

3.6 Estimated Timeline of the entire project –

Week 1(15-03-15 to 22-03-15)-
Learning the basic software required to implement the project-
Atmel Studio 6.0, Doxygen documentation generator, basics of Zig Bee communication through XCTU, etc.

Week 2(23-03-15 to 30-03-15)-
Work on development of the path that the bot will follow.
Generation of the set of co-ordinates to be given to the bot as taken input from the user.

PROTOTYPE SUBMISSION

Week 3(1-04-15 to 7-03-15)-
Work on actual motion of the bot along the path as calculated earlier. Also development of code for obstacle detection.

Week 4(8-04-15 to 15-04-15)-
Completion of work on obstacle avoiding and other miscellaneous things left. Integration of the complete code along with debugging and proper documentation.

3.7 Distribution of work into Modules-

Module 1- Code for generation of Path to be followed by the Bot which will also include implementing it using a canvas software like simplecpp or GLUT. (Can be characterized as TOUGH)
Will handled by – Meet Udeshi and Siddhant Garg

Module 2-The ZigBee interfacing of the Bot with the laptop. This will include how to send the coordinates obtained from the laptop to the Bot. (Can be characterized as MODERATE)
Will handled by – Prakirt Raj and Nishant Dhama

Module 3- Code for motion of the bot which will include how the bot will be following the path which is generated earlier.

(Can be characterized as MODERATE)
Will handled by – Prakirt Raj and Nishant Dhama

Module 4- Development of code for obstacle detection and also development of an alternate path to rejoin the original path.
(Can be characterized as MODERATE)
Will handled by – Meet Udeshi and Siddhant Garg

Module 5 – Miscellaneous work and documentation. This will include making of the presentations, reports, working videos of the bot, etc. (Can be characterized as EASY)
Will handled by – Meet Udeshi, Siddhant Garg, Prakirt Raj and Nishant Dhama

4. Quality control

Test Data

The Bot will traverse the path given to it through the Laptop (or android device) avoiding any obstacles in its path and will reach the final position as specified. Doing so it will also send back the locations of obstacles it encounters in this specific path.

5. Risk management

- The Bot may not detect the obstacle with a good enough accuracy (error due to the IR sharp sensors and proximity sensor).
- Once when the Bot deviates from the path given to it originally, it may never come back to the original path. There will always be an offset from the original path.
- The Bot may not have enough space to take turns (considering its dimensions) if it does not get the obstacle position accurately.
- There may be some delay in communication by the ZigBee module and the Bot.

6. References

- USB interface tutorial covering basic fundamentals Bluetooth (http://www.eeherald.com/section/design-guide/esmod14.html)
- An Introduction to Bluetooth Programming (http://people.csail.mit.edu/albert/bluez-intro/)
- E-Yantra (http://www.e-yantra.org)
- ZigBee Reference manual (https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf)
- Firebird V hardware and software manuals
- Wikipedia (www.wikipedia.com)