System Requirements Specification for : Smart Overtaking Bot

Abhishek Kabra	Kapil Dubey	Kanwal Prakash Singh	Sourabh Bhola
09005037	09005038	09005031	09005022

March 20, 2012

under the guidance of
Prof. Kavi Arya

Embedded and Real-Time Systems Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology
Bombay

Contents

1 Introduction

The main idea of this project is to develop an automated system where the bot is able to judge the speed of the vehicle and overtake it.We are writing the code for this system in Real time operating system(RTOS).The purpose of building this project on RTOS is because RTOS is an operating system intended to serve real-time application requests.A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task, this is the main reason all the application are moved to RTOS.
A real-time OS has an advanced algorithm for scheduling. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency; a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

1.1 Definitions, Acronyms and Abbreviations

- **I-bot :** Sense a vehicle in front of it, analyze its speed and then overtakes it with a greater speed.
- **Vehilce bot :** Keep moving at a constant speed in the direction of the I-bot in the inner circle.

1.2 References

- NeX robotics
- IAR Systems
- AVR on Ubuntu
- FireBird V Software Manual
- FireBird V Hardware Manual
- Spark V Software Manual
- Spark V Hardware Manual

2 Overall Description

The underlying idea of this project is to develop an automated system where the I-Bot is able to estimate the speed of the Vehicle-Bot and overtake it. We are planning to write the code for this system in Real time operating system(RTOS). The reason being RTOS is an operating system intended to serve real-time application requests, and that would be an inetgral part for the system we intend to build. A key characteristic of RTOS which is essential to our system is the level of its consistency concerning the amount of time it takes to accept and complete an application's task.

2.1 System Environment

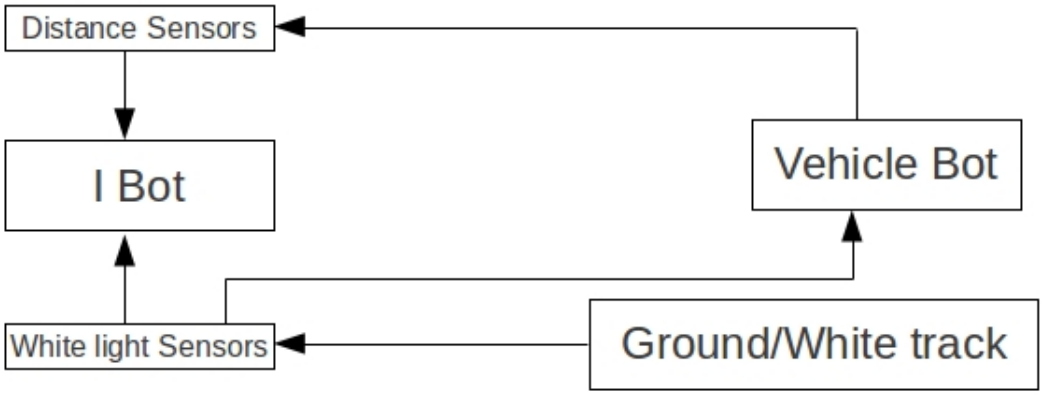


Figure 1: System Environment

The bot moves in an arena that has three tracks of white lines. The idea of using white lines is based on the fact that we would be using white line sensors for the movement of the bots on the tracks. The tracks are curved and make three parallel loops. The loops are used because of the limitations of a straight line overtaking as it reaches an end. Also, this way our project would be more close to the real life situation. Apart from this, the three track idea is based on our idea of choosing the shorter path while overtaking over a curve as we know the smaller the radius of curvature of the overtaking path the smaller would be the path to traverse while overtaking. But this also adds to the complexity involved in keeping control on the speed of the bot over the curved path, when the bot has to constantly monitor its own path so that it doesn't lose its track while overtaking. All the bots shall move in the middle track in the initial situation in uni-directional manner with I-Bot following the Vehicle-Bots. Also, we plan to include a number of Vehicle Bots (all SPARK bots) moving on the given tracks which will add to the complexity of the problem as the I-Bot has to constantly monitor if any Vehicle Bot is moving ahead of it and overtake it smartly, avoiding all kinds of accidents. The final feature that we add is the varying speeds of the Vehicle (SPARK) Bots, which would again pose a challenge to the safe overtaking by the I-Bot.

The I-bot will sense the Vehicle Bot moving ahead of itself using Sharp Sensors and would do Image Processing of the Vehicle Bot to judge its speed, the distance between itself anf the Vehicle Bot etc, and based on the calculations performed (like if the speed of the Vehicle Bot falls below a certain threshold speed, the distance between the two bots, the curvatur of the path it is currently moving on, whether or not it is straight etc, it would decide to overtake the Vehicle Bot safely and taking the optimal path in case of paths with curvatures). Also the I-bot regains its original track back once it overtakes the Vehicle-Bot.

Everything descried here is well shown in Figure 1 and 2.

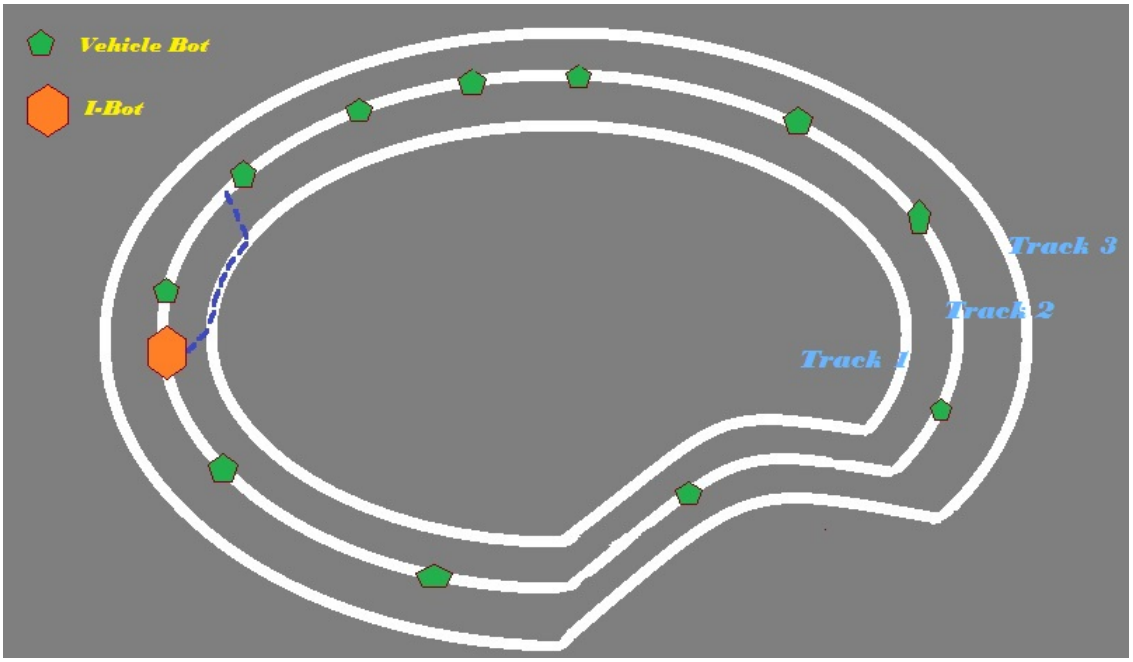


Figure 2: System Environment

2.2 Product Perspective

The functianlity of our Smart Overtaking Bot is in contiguity with the real life scenario. We have straight as well as curved paths on a road track which exactly match in our case. The freedom of overtaking as in unidirectional lanes has been approximated by us using three white track-lines. The varying speed of the Vehicle Bots further makes it nearer to the real life scenario. Above all, Overtaking is an 'intelligent' process as it involves a certain amount of calculation to be performed as accurate as they can be, and that too in real time, so as to avoid catastophic accidents.

We are building on the current scenario that has already been developed - Cruise Control and Adaptive Cruise Control, which enables a vehicle behind to slow down its speed if the vehicle ahead of it reduces its speed and thus help in traffic management.

The extension that we have provided to the existing idea of Adaptive Cruise Control can be used for Smart Racing, where the Bot taking decisions optimally while overtaking the bots coming in its way wins. It can also be used to design cars for the police for catching thieves. The USP of our product lies in its likeness with the real-life situation, and an idea that can be further built on optimization of the algorithm involved, while taking care of the dangers and complexities involved at the same time. The Image processing component involved in our project can also be enhanced upon for betterment.

2.3 Product Functions

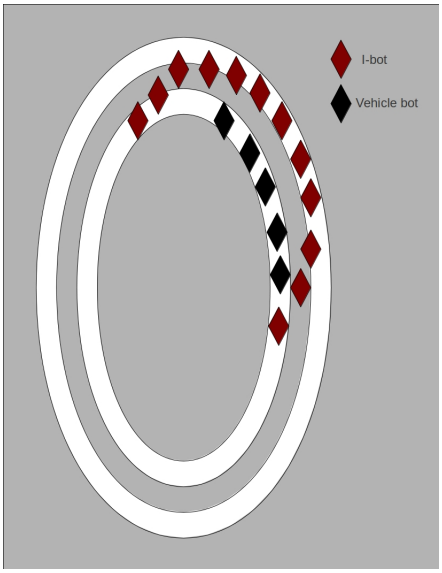
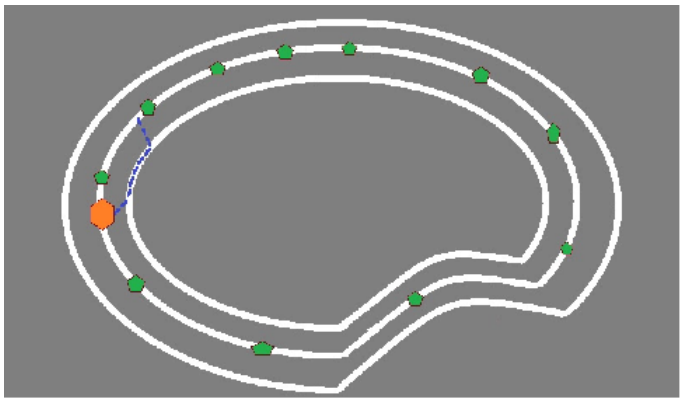
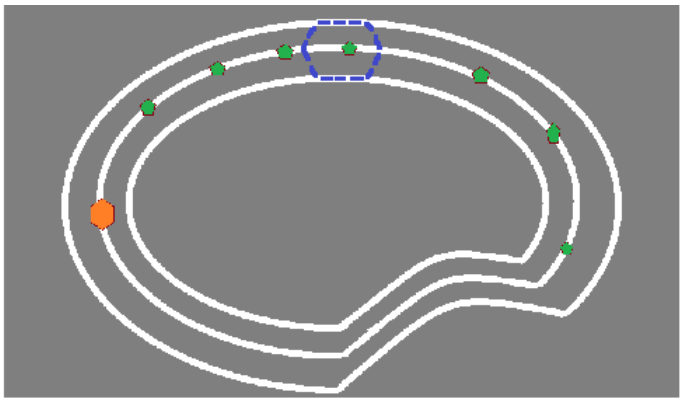


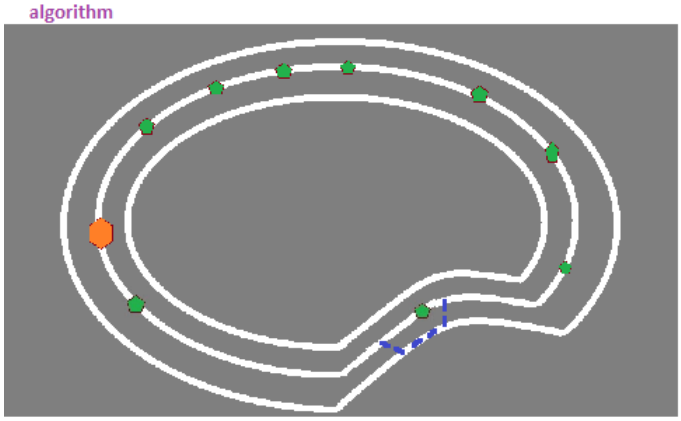
Figure 3: Robot Control Process



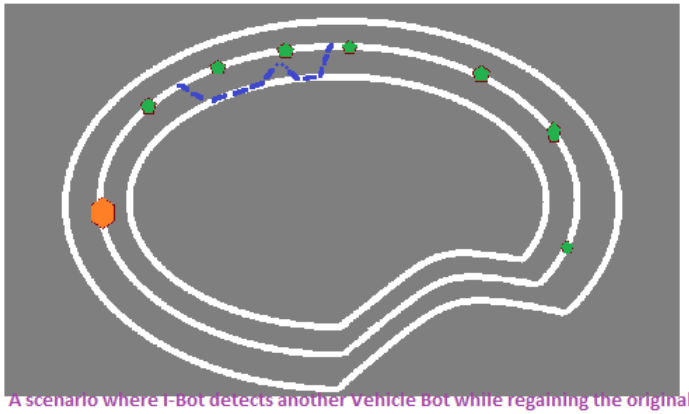
Overtaking over a curvature, I-Bot takes Track 1 to overtake based on its algorithm



Overtaking over a nearly linear path, I-Bot can take either of the two tracks, Track 1 or Track 3 for overtaking



Overtaking over another curvature, now I-Bot takes Track 3 for overtaking



A scenario where I-Bot detects another Vehicle Bot while regaining the original pa and thus moves back to the overtaking path, and again tries, this time successfully

Figure 4: Smart Path while overtaking

The bot that we are developing will be an automated overtaking vehicle. It will sense the vehicles on the track it is running and will move on the other track so as to overtake the vehicle bots in front. The product can be improved upon further by mounting a camera over it so that there is enough scope of human inputs on the bot control after getting inputs from the camera/sharp sensor.

2.4 Model

There will two concentric circular (or elliptical) tracks. The tracks are laid out using white line. All the tracks move on inner track. The track is changed only by the I-bot when it needs to overtake the vehicle bots. In this case, it moves on the outer track, overtakes and then gets back on the original track.

2.5 Constraints

- The I-Bots decision of overtaking should follow certain rules as specified in the Step-by-Step Movement
- The I-Bot should take care of the vehicles being present on the track to which it is changing its course of movement, in order to avoid accidents.

2.6 Assumptions and Dependencies

- The I-Bot is the Firebird Bot and the vehicle Bots are SPARK Bots.
- There are three tracks made with white color and the Vehicle Bots (SPARK Bots) and our I-Bot (Firebird Bot) are assumed to be moving on the central white line track.
- The I-Bot switches its track only (switching in accordance with the step by step movemnt specified) when it needs to overtake some Vehicle Bot moving ahead of it and gets back to the original track, after finding a free space for itself on the original track, thus avoiding the collisions.
- The I-Bot becomes aware of the curvature of the track depending on the speed of its tyres.

3 Details

3.1 Functionality

Our product using RTOS, enables all the caluclation involved in an 'intelligent' overtaking to be performed in real-time. Again, being in contiguity with the real-life scenario, it can be very well modelled on an actual highway traffic, Smart racing as explained in Product Perspective etc. Also, it can be further built by optimising the algorithm involved in overtaking, the precautions to be

taken while doing it to avoid accidents, etc. thus serving as a good start for a mega project in future.

3.2 Supportability

- The functions used in the c files should be well defined in the header field in RTOS.
- The code should be written in a modular fashion.
- Interfaces of each module should be properly defined.
- Standard naming conventions should be followed for variables and functions.
- In addition these variable and function names should be self explanatory.
- The code should be well commented.

3.3 Design Constraints

- The path of the vehicles is simulated in an almost circular area with white lines drawn .
- Vehicle bots and I-bot move in the inner circle and when the I-bot senses a vehicle bot ahead it goes to the outer path to overtake it.

3.4 Requirements

The system requires the following requirements for its functionality.

3.4.1 Hardware Requirements

- firebird vehicle bots
- spark bot
- sharp sensors

3.4.2 Software Requirements

- AVR-bootloader
- IAR workbench

4 Risk Management

- The vehicles assumed to be moving at constant speed may not be actually moving at constant speed. So, in the course of time, they may collide.
  Backup plan: There will be proximity sensors at the front of the vehicle bots also which will give it signals when there is some vehicle bot very close to it. In this case, the vehicle bot will stop for a while and then again start to move.

- The I-bot may get out of track while changing from inner track to outer track or from outer track to inner track.
  Backup plan: The I-bot will return if within a specified (calculated) time period, it has not been able to reach a new track (outer or inner).

- I-bot after overtaking while still on the outer track, starts returning and gets one other vehicle-bot which was moving just before the bot being overtaken.
  Backup plan: The I-bot has a sensor at its left to know whether there is any vehicle-bot to its left. This sensor will be used while returning back to the original track. It it senses that there is some bot on the original track, then I-bot keeps moving on the outer track till it senses that there is no vehicle bot to its left.