Autonomous Parking Garage System – A Roomba Implementation

by

Raymond C. Detiveaux, B.S.C.S

Report
Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MSE

The University of Texas at Austin
May 2008
Autonomous Parking Garage System – A Roomba Implementation

Approved by
Supervising Committee:

________________________
________________________
Dedication

I would like to dedicate this work to my parents, whose loving support has been invaluable and motivational to my academic pursuits.
Acknowledgements

I would like to acknowledge Todd E. Kurt for his excellent work on [1] as well as the open source community for their and Todd’s continued work on the RoombaComm Java API.

I would like to acknowledge Howard Chen for his documentation in [2], which was instrumental in configuring the development environment for this project.

March 10, 2008
Abstract

Autonomous Parking Garage System – A Roomba Implementation

Raymond C. Detiveaux, MS Software Engineering
The University of Texas at Austin, 2008

Supervisor: Christine Julien

From going to work to just spending a day of shopping, many commuters face the constant struggle of attempting to find parking. One solution that has become commonplace is that of the parking garage. However, this is not without its pitfalls; drivers must still navigate through differing levels and paths, all the while looking for a parking spot and avoiding the possibility of a collision with another vehicle. It is this very issue that this project will attempt to tackle - using a Roomba (more specifically, an iRobot Create) to model a vehicle, a solution will be implemented that will help to reduce the trouble and risks involved with normal, everyday parking within parking garages.
Table of Contents

List of Illustrations ...................................................................................... viii

1.0 Introduction ............................................................................................ 1

2.0 Literature Review .................................................................................. 3

3.0 Problem Specification .......................................................................... 6

4.0 Implementation .................................................................................... 7
  4.1 Architectural Overview ........................................................................ 7
  4.2 Configuration of the Roomba ............................................................. 9
  4.3 Configuration of the Virtual Walls ...................................................... 10
  4.4 Configuration of the Base Station .................................................... 13
  4.5 Code Implementation ......................................................................... 14

5.0 Analysis ............................................................................................... 16

6.0 Conclusion .......................................................................................... 19

Appendices .................................................................................................. 20
  Appendix A: Operational Application Code ............................................ 20
  Appendix B: Movement and Detection Algorithm ................................ 22

References .................................................................................................. 24

Biographical Vita ......................................................................................... 25
List of Illustrations

Illustration 1: Architectural Design ................................................................. 9
Illustration 2: Virtual Wall Modification ......................................................... 11
Illustration 3: Virtual Wall Placement ............................................................ 13
1.0 Introduction

In science fiction, there has long been the premise of a robot valet autonomously driving a vehicle around. Some would possibly question the practicality of such, however, the need is evident specifically with regards to assisting the elderly and/or those with disabilities. Then there are space concerns for parking with an increasing population, specifically in urban areas. Having identified two target areas for potential improvement, the idea of an autonomous parking garage system emerged. While there are fully robot controlled parking garage systems and other similar projects, few take into account both the potential human limitations as well as the economic and population concerns. This project will attempt to focus on both as the premise for the concept.

For the purpose of the project, the goal is to allow a vehicle to be parked independently of the driver. Additionally, as is the case for many of the newer fully robot controlled parking garage systems, the design of many pre-existing parking garages will need to be slightly altered to make benefit of the proposed implementation. No longer will there need to be ramps or an abundance of unused space.

Instead, imagine a one-level plane of parking spots within a parking garage. This one level would fill slots sequentially with each vehicle parked right behind one another. Now, in the interest of conserving space, rather than having pre-designated spot sizes, infrared transmitters would instead be used, and they could be moved dynamically to adjust to each vehicle’s dimensions. Then, the next line would open up and the routine would be repeated until that level was filled.

The intent of this report is to discuss this implementation of an autonomous parking garage system. Using a Roomba (more specifically, an iRobot Create) to model the vehicle, infrared transmitters and a base system will be used to create a centralized
parking garage system. The Roomba will then be able to communicate with the base station and park itself in the designated parking spot.

After doing so, there are concerns for vehicle retrieval, but some possible solutions have already been implemented to tackle this issue. Moreover, these solutions are trivial in comparison to eliminating the human factors required with parking itself. More discussion on this note will occur at a later point within this report.
2.0 Literature Review

As previously mentioned, this implementation will require communication between both the vehicle and the base station, as this will be a centralized parking system. Additionally, one of the primary motivators behind this implementation is that of human safety. Indeed, this concern has already received much attention and under such a schema, the idea of telematics has emerged [3]. Telematics is defined as the two-way communication between the automobile and base station. The drive behind this technology is to evolve the driving experience into one that is more fun, but more importantly, safer. Along these lines, the idea of an Autonomous Parking Garage System has been derived.

More along the lines of safety again, Automotion Parking Systems built a robot-parking garage in New York [4]. According to the manufacturers, the system boasts a complete impossibility that damage could ever occur, which would truly be a major feat and an unheard of safety record with regards to traditional garages.

This aforementioned safety is accomplished by the fact that the very technology that the implementation is built on is that of similar technology that is used for mechanical handling and document retrieval [5]. In this model, the driver is only required to move the vehicle into an entrance module where it is then transported to a parking slot by a robot trolley. Understandably, this simplifies the process for the driver, but the process of maneuvering around at least the base floor of the garage and placing the vehicle correctly within the trolley confines is still an issue.

This approach definitely has merit in its vehicle delivery to parking, and the included safety to the vehicle during the process. However, the human concern is still apparent. This project will attempt to alleviate all such human concern, but the aspect of
vehicle delivery, both to and from parking, is an amazing engineering feat that definitely has some potential application to this project.

In another pursuit, a system was designed where a robot would utilize an onboard GPS unit to locate the nearest parking facility, and then employ vision-based control to ultimately park the robot [6]. The GPS portion is an implementation that has been performed before, as well as other systems such as web sites that help to identify the closest available parking units.

While the GPS approach does attempt to focus on the same self-parking aspect of this project, it does not take into concern the economic and population concerns that are truly emerging. Rather, it focuses on pre-existing parking garage architectures, which do not fully utilize the available space, to locate and maneuver around. Along a similar line of thought, this seems to draw further concern when considering that of the newly emerging robot-controlled garages that operate with a lift system, and whether such a GPS/vision-based system could handle such dynamic variations. Ultimately, this system could be useful in locating garages regardless of their architecture, but for the purpose of autonomously parking the vehicle, it falls short of what this project is attempting to accomplish.

Speaking further on the automated parking aspect, another endeavor focused on developing a robot that could perform automated parallel parking [7]. The motivation behind this project stems from the automotive manufacturing industry from such manufacturers as BMW and Lexus, which have both initiated the concept of a “self-parking car” whereby a driver is still required to position the vehicle into location but then the automobile itself is capable of maneuvering into the “parked” state under automated control. The effort undertaken in this project was premised on expanding the automated parallel-parking concept, by providing additional automated parallel-parking
capabilities, such as self-discovery of parking spaces and coordinated avoidance of objects.

The key commonality between this approach and the project is the idea of self-parking. However, at this time, it is not believed that self-discovery is a feasible implementation in terms of safety. Without boundary controls, there is truly no way to prevent any type of discoloration or unexpected interference to prevent the vehicle from doing harm to itself, other vehicles and/or, more importantly, other people. This project will harness the ability of infrared transmitters along with a centralized base station to facilitate the parking in a much safer manner.

As the above implementations have all been traversed and examined, it can be seen that the primary motivators behind each project have been clearly different. Each either attempts to fulfill the need of the economic and population concerns or focuses on self-discovery and automated parking. Understandably, all of these approaches have been huge undertakings with much thought and emphasis on their designs. Yet, none of them attempt to focus on all of these aspects, and subsequently is where this project implementation differs from that of others. This project will focus on complete control of the vehicle as the primary facilitator of safety, and therefore the primary concern, as well as offer suggestions as to how the economic and population concerns could be addressed.
3.0 Problem Specification

The main problem at hand with parking garages is that of safety, both for the vehicles, the humans driving them, and the pedestrians that are walking within the parking garage. The second greatest concern is that of fully utilizing the available space as population increasingly advances. To continue with these primary motivating factors, a system will be modeled that will help to increase safeness by removing much of the human interaction and altogether eliminating the need for pedestrians within the various parking levels, except for the primary entrance location. Additionally, economic concerns such as space will be addressed by attempting to use all available space within the parking garage.

Although this implementation will not be deployed in an actual parking garage, the system will still be modeled in a similar fashion with suggestions on various architectural improvements to best utilize the proposed parking system. For the purpose of the project, the immediate goal is to allow a vehicle to be parked independently of the driver via a scaled-down model of the system.

Using a Roomba (more specifically, an iRobot Create) to model a vehicle, infrared transmitters are used to mark corresponding lots in a parking garage. Each lot appears in a linear sequence, and the gaps between the transmitters serve as the parking spots. The lot occupancy state is maintained by a central parking garage system, or base station, that the vehicle checks to determine what lots are available, makes a selection, and then moves to the designated lot. After arriving at the lot, the vehicle terminates movement as it enters the “parked” state.
4.0 Implementation

4.1 Architectural Overview

The primary objective of this project was to autonomously park a vehicle within its designated parking spot. In order to facilitate this implementation, it was decided to use a Roomba to emulate the vehicle. The robot should then be free to move in a linear path directly to the designated parking spot. In order to do this, it was determined that some type of emission signal would need to be used in order to identify to the robot that it had indeed entered the predetermined parking spot. With this intent, the Virtual Wall was decided upon to act as the separation markers as the Roomba hosts sensors that natively detect the infrared transmission from these devices. In this way, each parking spot could then be clearly defined by each division of infrared transmissions.

For the vehicle to know what parking spot it is going to, where it is currently located, or when it will stop, it is required for the Roomba to communicate with the base station. In order for the robot to communicate with the base station, it was decided to utilize the Bluetooth wireless protocol, and this was accomplished through the usage of a Bluetooth Adapter Module accessory for the robot.

The base station itself hosted a Bluetooth radio to facilitate the communication directly with the Roomba. Additionally, the base station was the core behind the operational aspect of the vehicle. The base station maintained the state of the robot in terms of both its sensor updates and its location. This was all performed through leveraging the Java programming language to develop code that would reside on the base station and establish a communication between itself and the Roomba to control the vehicle’s movement.
In a test scenario, the Roomba is placed to move in a linear fashion towards a series of similarly placed sequential Virtual Walls. The base station communicates directly with the robot as if the vehicle would just have entered the parking garage and mitigated the proper transaction to start the parking protocol. Once this communication has occurred, the Roomba would then move forward while it relayed sensor updates to the base station. The base station would then analyze these updates and send the vehicle a response regarding how to continue accordingly, based on the sensor analysis. Once in the predefined parking spot area as delimited by the Virtual Walls, the Roomba would then come to a stop as it entered the parking state according to the base station’s mapping of the robot’s state. Illustration 1 shows a detailed schematic of this process, along with the various elements involved.
Illustration 1: Architectural Design

4.2 Configuration of the Roomba

The specific robot chosen to model the vehicle was the iRobot Create Programmable Robot from the Roomba family of robots. This robot and model was primarily chosen for its open availability for use in this project as well as its default ability to provide out-of-the-box programming capabilities [8]. Additionally, the iRobot
Create provides an open-access payload bay, which provides an open platform for adding additional sensors, radios, robot-control devices and other 3rd party accessories.

This feature is what really opened the design up with the addition of BAM (Bluetooth Adapter Module) for iRobot Create. The BAM connects directly to this payload bay and provides wireless access via Bluetooth to control the iRobot Create from the base station. This wireless access is facilitated by means of the BAM as it provides a virtual serial port connection between the base station and the robot [9].

The next major component of the iRobot Create was its multitude of sensors, which actually numbers over 30 built-in sensors [8]. With this wide availability of sensors there is much that can be done for the iRobot Create to respond to both internal and external events, however, the main sensor that was of interest to this project was the Virtual Wall sensor. The iRobot Create hosts an Omnidirectional IR receiver that it uses to detect the Virtual Walls’s beam and halo [10]. This is very important and will be the subject of further discussion in the next section.

4.3 CONFIGURATION OF THE VIRTUAL WALLS

The usage of the iRobot Create provided the availability of using a sensor that was already configured to detect the presence of Virtual Walls. The iRobot Roomba Virtual Wall unit itself utilizes an infrared transmitter that can be used as either a homing beacon or as a confinement beam for the robot depending on the programming application [10]. Typically, these transmissions would occur from the Home Base (another optional accessory) in the form of a homing beacon to have the iRobot Create navigate back to “home” to recharge, or from the Virtual Wall as a confinement beam to prevent the robot from movement past the virtual barrier. However, in the case of this project, the Virtual Wall is used as a marking beacon, and since the robot is being controlled, it can move through such a virtual barrier.
Each Virtual Wall sends out an omnidirectional infrared transmission in the specified radius as selected on the device: 0-3’, 4’-7’ or 8’. There was an initial concern that the Roomba would detect the Virtual Wall before it actually arrived where the Virtual Wall was positioned. Therefore, a simple workaround was used to focus the infrared beam – electrical tape was used to block off the transmission except for directly in front of the Virtual Wall, as can be seen in Illustration 2.

Illustration 2: Virtual Wall Modification

Thus, when the Roomba moves forward, its Virtual Wall sensor only detects the IR transmission slightly prior to being directly in line with the Virtual Wall and will no longer detect the signal after it has moved slightly past the Virtual Wall.
Now that it was determined how to limit the detection of an individual Virtual Wall, there was still the need to differentiate between many Virtual Walls. The Virtual Walls not broadcasting any type of unique identifier confounded this. So it was determined that the Virtual Wall’s own ability to set the broadcast radius would be used to keep track of the Roomba’s current location rather than having to rely on pre-known boundary beacons (assuming there was a unique identifier to do so with).

In testing, it was found that a signal radius of 0-3’ would drop off pretty sharply after the 3’ interval to either direction of the Virtual Wall. Moreover, since the Virtual Wall sensor is positioned at the front of the Roomba, this means that the Roomba will detect the infrared transmission much sooner before arriving at the Virtual Wall. Although the electrical tape limits the transmission beam, it still sends it out in a slight cone that the omnidirectional sensor is able to pick up quickly due to its positioning on the robot. However, when the Roomba moves past the Virtual Wall, the sensor does not detect the infrared beam’s transmission for quite as long, since the Roomba’s front sensor moves out of the still residual cone effect quite promptly.

At this point, it has been defined how the Roomba should interact with one Virtual Wall, but given the detection overlap that exists due to the conical infrared transmission, more configurations were needed for the usage of multiple Virtual Walls. As mentioned previously, it was found in testing that a signal radius of 0-3’ would drop off pretty sharply after the 3’ interval to either direction of the Virtual Wall. Therefore, it was determined that each Virtual Wall needed to be placed slightly further than 3’ away from each other. A diagram of this configuration appears in Illustration 3.
On a final note, it should be mentioned that setting the distance between the Virtual Walls to be any less than the radius, as defined by the user-selectable option, will cause the Roomba to not properly be able to distinguish between the infrared transmissions, resulting in unexpected behavior.

4.4 Configuration of the Base Station

The base station in this project was simply a desktop computer with a Bluetooth adapter. As previously mentioned, the iRobot Create was mounted with the BAM to allow communication between the robot and the base station. Element Direct, the manufacturer of BAM, provides instructions on how to connect to BAM from the base station via a virtual serial connection over Bluetooth, and it is fairly straightforward. After some basic testing via a terminal program to ensure that communication was working and basic commands were being followed, the preparation for the actual code implementation was ready to begin.
This preparation phase was the most involved of all the various stages of the project. The two main components in configuring the development environment were the RoombaComm Java API [11] and the RXTX Java Communication API [12].

The RoombaComm Java API is an original implementation by Todd E. Kurt and others, and is the premise for the code development in his Roomba project undertakings [1]. Furthermore RoombaComm is an open source Java API library that is described as a library that wraps the ROI (Roomba Open Interface) protocol and allows the Roomba to be programmed without needing to be concerned with hexadecimal, bit manipulation, or any of the further details involved with the ROI [1].

The RXTX Java Communication API is provided by the open-source community [12]. RXTX is, in essence, an open-source implementation of the Sun Microsystems Java Communications API (CommAPI). The justification for the utilization of this API is that Java does not natively provide a way to talk to serial ports, and therefore the subsequent external library is required [1]. Moreover, many of the projects that make use of RoombaComm will also make use of the RoombaCommSerial class, which is a definite requirement with regards to this project implementation.

Now, configuring these two can be difficult depending on the resources and/or guides used. Luckily, there was an excellent reference that provided basic installation instructions along with the links to the required components [2]. In following these instructions, the development environment was configured and complete.

4.5 Code Implementation

Given the constraints of the previous sections, the Virtual Walls were used to mark the corresponding parking spots in the garage. In essence, each Virtual Wall is placed in a linear sequence and serves as a beacon to the Roomba that it will be entering a parking spot as soon as the infrared transmission is no longer detectable. The available
parking spots are represented as an array in which the spot is either taken or empty, and the Roomba will always choose the furthest spot available.

The array itself would be defined as lotmap = {0, 0, 0} assuming that there were three Virtual Walls, which would represent the three different parking spots, and that none of the spots were taken. This should be noted as being a valid state whereas {1, 0, 1} is an invalid state. The state of {1, 0, 1} would imply that a vehicle was never parked in the second spot, and thus would essentially be lost. However, the states of {1, 0, 0} and {1, 1, 0} would be valid as this could mean that parking spots towards the end were already vacated. To further refine this, the Roomba will always try to fill all parking spots in a sequential order, and when the vehicles would depart, they would do so in the order that they appear in the line of parking spots.

As to the actual implementation, most has already been covered in terms of the system configuration, but it has not yet been discussed what the robot is actually doing. First, the Roomba determines the lot availability and selects the furthest non-blocked, non-empty spot. Then, the Roomba updates its sensors and begins its journey towards the destination. As it passes each Virtual Wall, the Roomba increments a counter and determines if it is has entered said destination. If it has, then it stops and enters the “parked” state. Otherwise, it updates its sensors, moves forward and repeats this process until it has finally migrated to the designated parking spot.
5.0 Analysis

By following the documented steps as defined in this report, the project was successfully deployed. It should be noted that there are a few assumptions that influence the system and its design. First, it is assumed that the robot is positioned in a straight line parallel to that of the linear sequence of Virtual Walls, which are positioned linearly to one another in a straight line. Next, it is assumed that there are no barriers or curves that the Roomba would need to avoid or follow in order to reach the final destination. Finally, it is assumed that only one robot is being parked at all times.

To tackle the first concern, the key issue is more about moving the Virtual Walls from their fixed position relative to the Roomba’s expected path. In order to allow for such to occur, it was necessary to implement an emitter system that involved a signal that broadcasted a unique identifier as well, within its own transmission. In this regard, the chance of transmissions intersecting would eliminate the possibility of unexpected robot behavior, as each separate emission beam would be discernible from the other. The only additional challenge that this would impose would be that of maintaining an accurate mapping of each Virtual Wall’s location along with the unique identifier associated with each.

Again, moving away from the linear approach, the boundaries are now the key motivational factors when taking into consideration barriers or curves. While there are several approaches, most revolve around the idea of pattern- or edge-detection. Whether it be a physical boundary [2] or, as in the self-parking car approach [7], path lines on the pavement, it will still require a specialized sensor to keep the vehicle within this area until reaching the pre-designated parking spot. While these do provide solutions, they
also provide additional upkeep in the form of delineating markers that can all deteriorate to varying degrees.

As for parking multiple vehicles, this becomes a much simpler task assuming that the architectural design has been firmly tested and adhered to. In essence, the base station would need much more advanced operational code to control the operation of several vehicles. The primary suggestion to accomplish this goal would be develop a distributed system with several base stations. Each vehicle would then make entry into the parking garage and establish communication with one of the available base stations. Then, the base station would treat the available parking spots as a critical code section. The base station would seek entry into the critical section from amongst the other base stations and either enter into the inventory directly and thus reserving the first available parking spot to the vehicle, or it would place the request into the queue and then gain access to the critical section upon its turn, again proceeding to reserve the first available parking spot to the vehicle. Under this model, there would be only one vehicle moving forward towards its destination at any given exact moment in time. All other requests would be slightly staggered due to the necessity of gaining access to the parking spot inventory, which would reduce vehicles into entering a queuing system by which accidents could be prevented.

For the remainder of the implementation, there is still some question about the technologies and requirements. In terms of communication, it is not felt that Bluetooth would be scalable as the protocol only allows for an approximate communication distance between 1 and 100 meters depending on the class of the Bluetooth device (Class 1 having the most distance and Class 3 having the least distance). This would require rewriting of the application code along with fitting each base station and vehicle with the appropriate communication device, which in this case, Wi-Fi would serve as a very likely
candidate. Moreover, as discussed previously, each vehicle would still require two sensors to continue under a “real-world” scenario. The first sensor would be responsible for detecting the infrared transmissions along with their associated unique identifier. The second sensor would be responsible for maneuvering the vehicle around autonomously to the lot destination should the vehicle movement course not be configured in a linear fashion, as was the case in this project. Of course, in the case of any pattern- or edge-detection schema, this would also likely require the vehicle to be retrofitted with an additional bumper plate and embedded sensor within to reduce vehicle damage. While there are other possible implementations that would not require this such as in the self-parking car project, this is still a demand that must be placed on the vehicle. Therefore, going forward with a “real-world” implementation would require that every vehicle wishing to be serviced in a parking garage system under this paradigm would require at least one additional radio for communication, at least two additional sensors for operational and state mechanics, and potentially up to one enhanced sensor bumper plate.

Regardless of these assumptions and their subsequent analysis, as stated above, the project was still successful. It accomplished the goal of having the Roomba find an available parking spot, if one exists, and then moving itself to that position where it ceases operation and enters the “parked” state.
6.0 Conclusion

The main finding of this project was that a fully autonomous parking garage system could indeed be implemented. In bringing all of the components together to look at the big picture, it can be seen that the method used to park the vehicle in this project does so in a linear fashion, which is not how conventional parking garages work. Therefore, it would be suggested that the system created here would be much more useful in terms of augmenting the fully controlled robot lift parking garage system [5].

A potential drawback to this approach is that waits may occur for the vehicle driver and its occupants, especially during peak periods. This wait is primarily due to a single bottleneck in the system – loading passengers and luggage only occurs at the entrance and exit location rather than at the parked vehicle itself, which can cause blockages [5]. The proposition here is to expand the system implemented in this project to autonomously park the cars within the trolley confines while also providing a queue-based system to prevent this wait time from truly impacting drivers.

Moreover, automotive manufacturers are increasingly adding more and more safety features that assist with the driving experience as well as fully autonomous components such as self-parking [7]. With the inclusion of such abilities being a basic component of the vehicle, then communicating directly with the vehicle to perform a single autonomous operation at the direction of the centralized parking garage system would be a much simpler development endeavor. Not only that, but the system would also have the ability to handle the unexpected events through such mechanisms as collision and obstacle avoidance and advanced self-parking to allow for differing entry approaches to different trolley mechanisms in automated parking garage systems.
Appendices

APPENDIX A: OPERATIONAL APPLICATION CODE

The following code snippet is the base for the operational application code:

```java
public class Test {
    static String usage =
        "Usage: \n"
        +
        "  roombacomm.Drive <serialportname> [options]\n" +
        "where [options] can be one or more of:\n" +
        "  -debug   -- turn on debug output\n" +
        "  -hhandshake  -- use hardware-handshaking, for Windows Bluetooth\n"
        +
        "\n";
    static boolean debug = false;
    static boolean hwhandshake = false;

    public static void main(String[] args) {
        if(args.length < 1) {
            System.out.println(usage);
            System.exit(0);
        }

        for(int i=1; i < args.length; i++) {
            if(args[i].endsWith("debug")) {
                debug = true;
            } else if(args[i].endsWith("hhandshake")) {
                hhandshake = true;
            }
        }

        roombacomm.RoombaCommSerial roombacomm = new roombacomm.RoombaCommSerial();
        roombacomm.debug = debug;
        roombacomm.waitForDSR = hhandshake;

        if(!roombacomm.connect(portname)) {
            System.out.println("\nCould not connect to: " + portname);
        }
    }
}
```
System.exit(1);
}
System.out.println("\nRoomba startup.");
roombacomm.startup();
roombacomm.control();
roombacomm.pause(100);
roombacomm.updateSensors();
System.out.println("Press return to exit.\n");
// BEGIN: Changeable block

// END: Changeable block
roombacomm.stop();
System.out.print("\nDisconnecting . . . ");
roombacomm.disconnect();
System.out.println("Done!");
}
}

The comments are basically to indicate where the functional code for controlling the Roomba should go. Basically, the skeleton implementation between these two comments would be something as follows with the appropriate code added in as necessary:

while(!done)
{
    // Code implementation here
    done = keyIsPressed();
}
APPENDIX B: MOVEMENT AND DETECTION ALGORITHM

This is the basic movement and detection algorithm that was placed within this operational construct as described in Appendix A:

```java
while (!done)
{
    roombacomm.updateSensors();
    vw = roombacomm.virtual_wall();

    while (vw > 0)
    {
        if (flag == 0)
        {
            ++lotnum;
            flag = 1;
        }

        roombacomm.goStraightAt(speed);
        roombacomm.updateSensors();
        vw = roombacomm.virtual_wall();
    }

    if (flag == 1)
    {
        System.out.println("Now in Lot #" + lotnum + ".");
    }

    if (lotnum == destlot)
    {
        System.out.println("Now in the destination lot!!!");
        break;
    }
    else
    {
        if (flag == 1)
        {
            System.out.println("Now moving to Lot #" + (lotnum + 1) + ".");
            flag = 0;
        }
    }

    roombacomm.goStraightAt(speed);
}
At this point in the code, the Roomba already knows the destination lot and just has to continue through the sequential Virtual Wall infrared transmissions until it finds said lot.
References


Biographical Vita

Raymond C. Detiveaux was born in Houston, Texas on June 12, 1980, the son of Abigail Ava Sager and Raymond Joseph Detiveaux. After completing his work at A.C. Jones High School, Beeville, Texas, in 1998, he entered Texas A&M University-Kingsville in Kingsville, Texas. He received the degree of Bachelor of Science from Texas A&M University-Kingsville in May of 2002. During the next few years, he was employed as an Oracle Analyst at Linebarger Goggan Blair & Sampson, LLP. In December 2003, he accepted the position of Software Engineer with IBM, where he is currently employed. In January 2006, he entered The Graduate School at The University of Texas.

Permanent address: 15109 Natural Spring Way, Austin, Texas 78728
This Report was typed by Raymond C. Detiveaux.