Autonomous Wheelchair Navigation Prototype with an Arduino Robot

Anna Shafer, Michael Turney, Francisco Ruiz, Justin Mabon, Michael Nooner, Yu Sun, Vamsi Paruchuri
Dept. of Computer Science, University of Central Arkansas, Conway, AR 72034, USA

Abstract - *Navigating through a large and complicated hospital can be difficult to most people, especially to those elderly and/or disabled patients. To help patients more efficiently while reducing the manpower, in this research, we have proposed and developed an Autonomous Wheelchair Navigation Prototype with an Arduino robot for hospital navigation. With a user-friend interface, the proposed prototype is able to determine the optimal path to find locations accurately and can successfully control robots' movement during the navigation. Thus, it can remove the need to learn the ins and outs of hospitals and improve the quality of life for its users. Our prototype for this system has shown good preliminary results and is looking towards a bright future.*

Keywords: Arduino, Robot, Wheelchair, Autonomous, Navigation

1. Introduction

1.1 Background

The United States Census Bureau released the status of people with disabilities in July of 2012. Over 3.5 million people use a wheelchair to assist with mobility throughout their daily lives. Depending upon the severity and type of disability, mobility and independence can be a challenge. This issue is compounded by the aging population of the United States. The post World II baby boom of 1946-1964 flooded the United States alone with 75 million births. The Baby Boomers are beginning to hit the 65 and older mark which will cause an influx of patients being admitted into hospitals. As such health care resources in the US are not prepared to meet such a rapid increase in elderly patients. These patients will require more and more medical care. The situation is already taking a toll on health care resources, requiring more health care personnel to accommodate the rising number of elderly patients [1].

The draw on human resources in the US as a result of this population imbalance is uneconomical. In comparison, Japan is ageing faster than any other country in history, with vast consequences for its economy and society [2]. They have, however, proposed a solution to some of the issues that we are now beginning to face. One such problem is the navigation of hospitals by elderly patients.

Navigating large hospitals can be difficult for patients and arduous for the caretakers of elderly patients. Without assistance, hospital navigation can be difficult or impossible for them. The Autonomous Wheelchair Prototype is the solution to this problem.

1.2 Related Work

Todays technological advances have opened many doors for those with physical impairments. However, current technologies are still lacking. Manual wheelchairs are still the standard in hospital settings, requiring a patient to be in good physical condition and have knowledge of the layout in order to navigate or have assistance from health care personnel. Several solutions to this problem have been proposed.

1.2.1 MICA

The Mobile Internet Connected Assistant (MICA), developed by the Lulea University of Technology in Sweden, allows users to operate a wheelchair with movement of the head, voice commands, or fully autonomously [3]. It is designed to be controlled remotely, over the internet, or by the user himself. Where the MICA suffers though, is its lack of pathfinding. The only autonomous navigation the MICA is capable of voice recognition which requires a user to dictate very specific commands. A user is required to be able to speak or make head movements to navigate this device and to have a knowledge of the environment in order to direct it.

1.2.2 RobChair

The RobChair navigation system was developed at the University of Coimbra, Portugal. The system is designed to assist quadriplegic or simultaneously blind and paraplegic people with their mobility and navigation in domestic environments [4]. The system is voice activated and is equipped with obstacle avoidance to allow for general navigation commands to *go right* or *go left.* However, this system lacks any knowledge about its surroundings. It requires the user to have a thorough knowledge of their environment and be able to navigate to their destination.

1.2.3 Aviator

The Aviator is a wheelchair designed by Hung Nguyen and his team at the Centre for Health Technologies at the University of Technology, Sydney. This hands-free

wheelchair uses an electroencephalography (EEG) to read and translate brain signals into navigational commands for the wheelchair [5]. While the wheelchair can navigate by thoughts alone and uses cameras to avoid obstacles, it is very expensive and made only for patients who are severely physically handicapped. It also requires the user to have a prior knowledge of their environment.

1.3 Motivation and Solution

Hospitals are huge and difficult to navigate. The long hallways and labyrinthine passageways are problematic for patients regardless of how many times they may have visited. Patients often have to rely on others to help them navigate, either by physically pushing them in a wheelchair or by guiding them to their destination.

Our proposed autonomous wheelchair navigation system will be able to transport a patient to their destination with only the push of a few buttons. This system will use RFID tags strategically embedded throughout the hospital to provide orientation information to the wheelchair device. Our system takes advantage of the static nature of hospitals by allowing previously populated floor maps of hospitals to be downloaded to the device at any time, allowing the device to navigate to any destination choice without requiring prior knowledge of the environment's layout from the user. RFID tags will be used because they are cheap, readily available, and will provide more accurate feedback than a GPS in this indoor setting.

Any patient will be able to use this system. It is designed to remove the need of learning hospital layouts. By allowing technology to assist in this way, better management of human and monetary resources will be possible. This wheelchair navigation system will save money for health care administrations while accommodating the needs of a wider range of patients than is possible with only a manual wheelchair.

The rest of this paper is organized as follows. Section 2 describes the overall system structure. Section 3 details the prototype's components. Section 4 discusses the initial results and Section 5 concludes the paper.

2. System Overview and Description

Fig. 1 illustrates the system overview. It starts with the Administrative UI (AUI), where a hospital administrator can create maps of a hospital. He can build maps to mimic floor plans and save them to an online repository, which contains all the floors and buildings of a hospital along with a database that links room locations and room numbers together. The database also links RFID values and tag numbers, so the admin only has to memorize tag numbers but not the 12 digit hexadecimal RFID values.

When a user sits in their wheelchair, the map of the floor is downloaded to the device. When the user inputs the room number they wish to go to, the wheelchair will then scan their current location and generate the optimal path to their destination. It will then send directions to the wheelchair's motors directing the wheelchair. When the wheelchair drives over an RFID card, it will compare the value with the virtual map. If the RFID card matches then the wheelchair is in the correct location and is given another direction. If the values don't match the wheelchair will update the path with the new location. Once the wheelchair reads the RFID card of its destination the chair will stop.

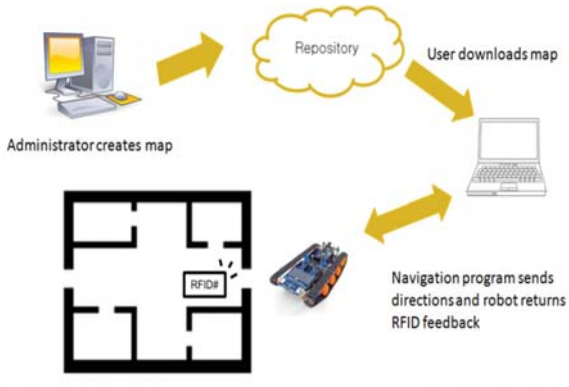

Fig. 1: Diagram of robotic wheelchair simulation

3. Proposed Prototype Components

The prototype wheelchair navigation system is broken up into four modules (Fig.2): the Administrative UI, the Software Keypad, the Navigational algorithm, and the Robot. The Administrative UI is used to create the maps of hospital floors. The Software Keypad is used to retrieve the room locations from the database. The location is sent to the Navigational Algorithm which determines the optimal path. The algorithm sends instructions to the Robot. The Robot moves according to the instructions and sends feedback to the algorithm to update its position.

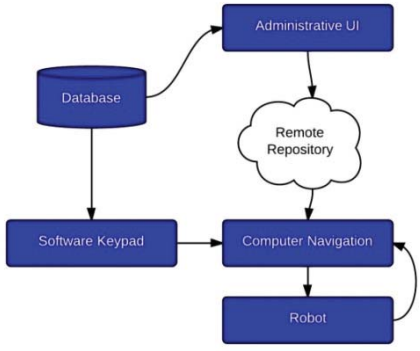

Fig. 2: Prototype wheelchair system flow

3.1 Administrative UI

The Administrative UI is designed in three stages: generate, build, and save. The administrator must first generate a map of a set size. Once the map is generated, the

administrator can then start to build the map. By using a toolbar of pre-defined shapes, the admin can reconstruct most floor plans. The admin is able to click the shape in the toolbar and place it in the map space. Doors and RFID-embedded tiles require the admin to input an RFID tag, which is a simple number assigned to each tag. During and after the building stage, the admin can save their work. The admin can then load the map at any time to continue work. When the map is finished and saved, it will then be in the online repository and it is ready to be used.

The AUI (Fig. 3) is created using JavaScript, PhP and HTML. The HTML is used to create a canvas for the map and toolbar to be drawn in. The JavaScript populates the canvases with the map and toolbar. The JavaScript is then used to draw the map by the admin clicking the tile in the toolbar and then clicking a location in the map. The load function uses JavaScript to open up a text file and parse the contents to find the value of each position in the map. The PhP is used to save the map. It also converts the RFID tag the admin inputted into the 12 digit hexadecimal RFID code. The PhP also finds the rooms in the map and stores the location of the room in a database. This database is used later by the software keypad to retrieve end-point locations.

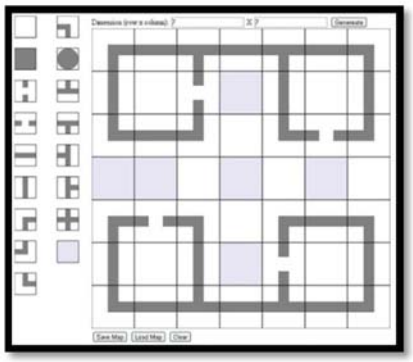

Fig. 3: GUI example map

From the Administrative UI Web Page, the admin can generate a map and start drawing. Fig. 4 shows the three main functions and how they are called. The Generate function will create an empty map for the admin to draw in. At this point the admin can then click around on the web page. Depending on where the admin clicks will determine which function is called. If the admin clicks in the toolbar, selectShape() is called to load a function into a variable. If the admin clicks inside the map space, drawShape() will be called.

The UI contains an array of functions. These functions are used to draw the many different shapes. When the user clicks in the toolbar, that function from the array is loaded into a variable. When the user clicks inside the map, the function in the variable is called. This lets the user draw many shapes in the map without having a large if-else chain to slow it down. The map is a 2d array that contains strings. The value of the string can range from "0" to "16" which represent the possible shapes. Values of "2", "3", and "16" all contain extra information. "2" and "3" are doors and must contain the RFID value in the floor tile and the room number. "16" is the RFID embedded tile and as such contains the RFID value. An example of a door tile is "3;2;101" which means a door to room 101 has the RFID tag 2. An RFID embedded tile would look like "16;4" which means the tile has the RFID tag 4.

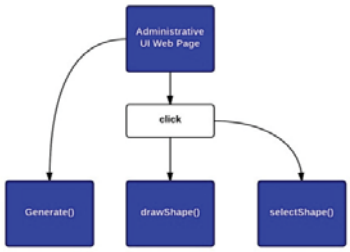

Fig. 4: Three Main Functions

When the user is done and wants to save, the save function is called. This function sends the 2D array representing the map to a php script. This script will go through the array and write to a text file the contents. It will place the location of the tile and its value. A value of 'w' means wall, a value of 'e' means empty, no value represents an empty space, and anything else is the 12 digit hexadecimal value. When the program detects there will be a 12 digit hexadecimal value, it will connect to a database that pairs the hexadecimal value with the RFID tag number. This allows the user to not have to memorize so much. An example of an entry in the text file is "2 2 45DCA0112385" which means that position 2,2 has that RFID value while "2 3 w" means there is a wall at 2,3. When the text file is completed, the user can then operate the software keypad to find room locations.

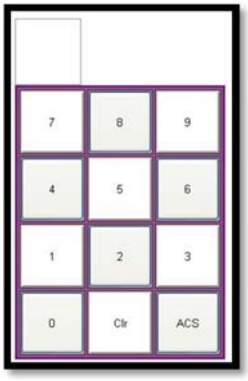

Fig. 5: Software Keypad example

3.2 Software Keypad

Our keypad (Fig.5), programmed in HTML and Javascript, contains mostly numbers and has twelve buttons, which is easy for users. When a patient uses our Autonomous Wheelchair Prototype, they would press on the keypad the room number they wish to go to. The keypad sends the room number to the remote map server. That room

number is looked up in the database and the database returns the location and saves that location into a text file. The robot then navigates to the destination that the patient wanted to go to.

3.3 A star Algorithm

The A star (A*) algorithm is standard in navigation. It finds the optimal path between two points in a known map which consist of points or nodes. These nodes are spread out in the map and serve as start and end points as well as all points in between. Each node also contains priority and level values. The priority value is equal to the distance it is from the end point added to its level. The level of a node is how far away the node is from the starting point. So a priority of 15 means that the node is 15 units away from the end point, while a level of 11 means this node is 11 units away from the start point.

The standard A* algorithm is not suitable for our project, so we had to make some revisions. One of those revisions is on the node structure. The node structure used in our A* algorithm contains six properties. These properties are: the X and Y coordinates of the node; the level and priority of the node; the RFID value of the node, and the direction that the node is pointing in. The direction is a value between zero and seven. Zero represents map east and each increment represents another direction in a clockwise direction such that seven will represent map north-east (see Fig. 6). This value is updated to show where the next node in the path is located. The major steps of the revised A* algorithm are: (1) Begin with the starting Node; (2) Search each neighbor node of the selected node. Start to the east of the node and continue clockwise until all eight directions are checked; (3) For each neighboring node, check to see if they are a wall node or have been visited before; (4) If the node is not a wall node and has not been visited before, place the node in a possible path queue; (5) If the node in the queue has been considered before, compare the two priorities of the paths; (6) Update the nodes priority and direction based on the lowest priority; (7) Place the nodes in a stack; (8) Select a node from the stack and go back to step 2; (9) Once the stack is empty or the end node is found backtrack using the direction on the nodes; (10) This will give the correct path.

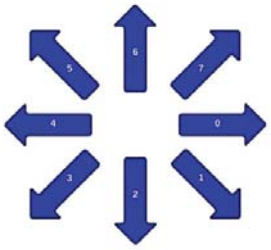

Fig. 6: Eight directions & numbers

The text file from the Administrative UI is required for the A* algorithm. Before the path can be generated, the program has to load the map into memory. It will parse through the text file and generate nodes based on the map information. When the path is found, the algorithm will return a string of directions. It will then send one direction at a time to the robot. These directions will indicate which way the robot should be heading. It follows the same numbering pattern from before where zero is map east. When a direction is sent to the robot it will wait for the robot to send back a 12 digit hexadecimal value. The program will take this value and check to see if the robot is in the correct spot. If it is, the program will send the next instruction. If the robot is off course, the program will update the path with the new starting location and send instructions to the robot.

3.4 Hardware

The hardware platform used in this initial prototype is based on the DFRobotshop Rover shown in Fig. 7. This robotic kit is constructed around an Arduino Uno microprocessor and its printed circuit board (PCB) which supports the control of the robot's motors, sensors, input and output ports, as well as communication with an external computer via mini USB. Our application takes advantage of the motor controller electronics as well as multiple serial lines. The Arduino microprocessor can be programmed using open source Arduino libraries adapted with C++.

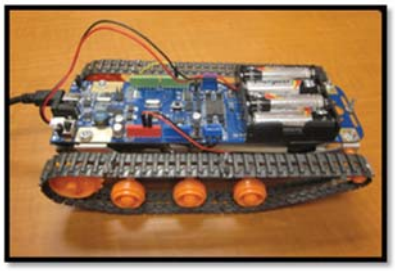

Fig.7: Assembled DFRobotshop Rover

The embedded motor controller chip on the PCB board is used to send alternating signals to the motors corresponding to the desired speed and direction. The robot is directed using a relative positioning algorithm that translates directional navigation commands from the computer into right and left turns relative to the robot's current position. After the command is translated, it is amplified by a factor determined by the robot's terrain and other variables such as the currently supplied power.

The prototype also utilizes a radio-frequency identification (RFID) reader (Fig. 8) as position feedback for the navigational algorithm. This RFID reader operates on a 125 kHz frequency allowing it to read standard electromagnetic card tags (Fig. 9). All communication with the RFID reader is performed through a serial line connected to the microprocessor. The tag numbers are read from the serial stream when they are detected. The tags consist of 16 total bytes in the format shown below. The RFID reader is connected to the PCB as shown in Fig. 10.

[start of text] – [12 bytes of hex] – [new line] – [carriage return] – [end of text]

Fig.8: RFID reader

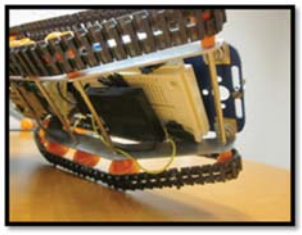

Fig.9: RFID reader on robot

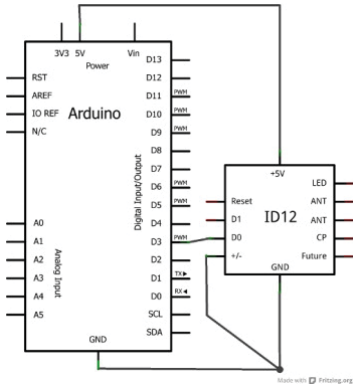

Fig.10: Wiring between RFID reader& Arduino

The robot is only capable of communicating through one serial line at a time. In addition, the USB cable is the communication line for navigational commands to be sent to the robot and the RFID numbers to be sent back to the computer. Our program uses the Windows API to open a COM (communication) port on the computer to receive the information from the microprocessor. All sending and receiving serial pairs must use the same baud rate throughout the communication process. The serial stream read by the program is error prone; therefore, it is necessary to add error correction to the serial processing. We implemented a basic error correction into our prototype; however, it is still possible for enough data to be lost that the only recovery option is an error message.

Fig. 11 illustrates the communication flow throughout the program. A communication handshake is performed at the beginning to verify that the synchronization has taken place. It lights up an LED to confirm that the handshake was successful and the robot is ready to begin communication with the computer. Then, RFID numbers are sent to the computer for position feedback and they receive direction commands in return until the destination is reached.

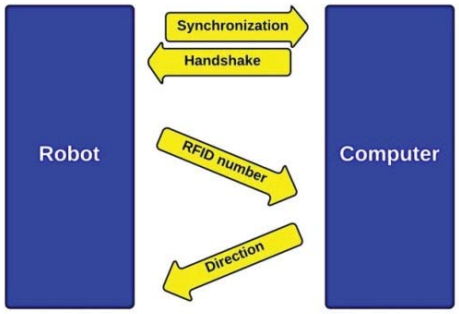

Fig. 11: Communication between robot & computer

4. Experimental Results

We tested our prototype of the autonomous wheelchair system according to two metrics. In order for the system to be successful, it must be able to quickly and accurately deliver the patients to their destinations. Thus, the navigational algorithm must be able to generate the path in a timely manner, and the robot must be able to reach the destination accurately 100% of the time.

We performed a worst case scenario test to determine whether the navigational implementation was timely. We created a random map with 5600 nodes. Considering hospitals can be as large as 500,000 square feet and can contain up to ten floors. Each floor would be around 50,000 square feet. Since a single node can represent a doorway which is about three feet standard. This would bring a representation of a hospital floor to a 75 nodes by 75 nodes map. Based on estimations of real world applications, RFID tags accounted for 24% of those nodes and 25% of them were designated as walls. The algorithm to generate the path was timed by subtracting the total milliseconds at the time that it started calculating the path from the total milliseconds that had occurred when the path was done generating. The path was generated, on average, in 840 milliseconds. The average human reaction time to change is around 240 milliseconds. So very shortly after the average human would react to entering their destination, the chair will already have the path generated.

Since the map generated is randomized, it is disorganized and contains many more possible paths than in reality. There are very few intersections in hospitals where there are eight possible ways to go, but since the map is randomized this situation shows up much more often. In a map of a real hospital with optimized RFID placement, the time to traverse the map and find the optimal path will be greatly lowered. Even in the worst case, it still takes less

than one second for the path to be generated and the wheelchair to start moving. When tested on more suitable rooms for the prototype, the path finding time was greatly reduced. For a 7x7 room the path was found in less than 16 milliseconds every run.

In addition to speed, accuracy of the path that was generated and the ability of the robot to follow that path was measured. Measuring how often the robot was able to correctly navigate to the destination was difficult to quantify due to the imprecise nature of the hardware used. To account for these dependencies in our testing, we divided the measurement of the correctness of the navigational algorithm's path generation from the actual hardware's response when designing our tests.

We tested our prototype three times on each of three different maps. The maps were designed to test the ability of the algorithm to correctly find the most efficient path in several different situations and the ability of the RFID reader to provide accurate positioning feedback. For each of the nine test runs, the number of correctly read RFID tags was divided by the total number of tags to be read for the generated path. These ratios are listed in Table 1.

Table 1: Ratio of correctly read to total RFID tags

	Test 1	Test 2	Test 3
Test Map 1 (4 rooms)	0.64	1.00	0.86
Test Map 2 (2 rooms)	1.00	0.89	0.88
Test Map 3 (1 room)	1.00	1.00	0.80

The average for each test map was then found. The average RFID tag reading accuracy for test maps one, two, and three was 0.800, 0.913, 0.923, respectively. The robot's overall RFID tag reading accuracy was calculated by taking the average of these three numbers, yielding an accuracy rate of 87.9%.

The results gathered from our tests so far, support further investigation into this wheelchair navigation system. On nine different tests, requiring the navigational algorithm to calculate different paths, the average run time was less than 16 ms. The time to generate a path in a worst case scenario was still less than a second. In the test cases, the accuracy of the robot was limited only by the hardware. The robot successfully navigated to its destination every time, unless the hardware failed, illustrating the potential of the system.

5. Conclusion

Navigating hospitals is a tedious and demanding task for patients and caretakers. With an increasing number of people aging, there will be a greater demand for health care resources in the near future. To alleviate the burden on health care, an autonomous wheelchair navigation system is very crucial.

We have investigated and developed a prototype for this system. Our Autonomous Wheelchair Navigation prototype was a success. It is able to navigate to an end point following the optimal path. The prototype was built with cheap, off the shelf components. In addition, it is able to generate an optimal path to its destination, which can autonomously traverse to that destination. The prototype successfully simulates the entire system by supplying the platform for an administrator to generate floor maps and save them to an online repository, a user download those maps to the prototype, and letting the robot travel through the map.

The core functionality of prototype is in place, but there are improvements to be made in the future. Bundling floors together in the administrative UI will reduce load times between multi-floor navigation. Adding additional sensors to the robot will help with turning and timing issues along with collision detection. Our initial prototype simulates an autonomous wheelchair hospital navigation system and demonstrates the great potential such a system has to assist our growing elderly population.

6. Acknowledgements

This research was partially supported by NSF Award# 1062838: "REU Site: HIT@UCA: Applied Research in Health Information Technology."

7. References

[1] "Chance of Becoming Disabled - Council for Disability Awareness",http://www.disabilitycanhappen.org/chances_disability/disability_stats.asp.

[2] "Into the unknown | The Economist", http://www.economist.com/node/17492860

[3] M. Hanlon, "The autonomous wheelchair raises the promise of assistive mobile robots," *Gizmag New & Emerging Tech. News*. http://www.gizmag.com/go/6626/, Dec. 16, 2006.

[4] G. Pires, N. Honório, C. Lopes, U. Nunes, A. T Almeida, "Autonomous Wheelchair for Disabled People", *Proc. of IEEE International Symposium on Industrial Electronics,* Guimarães, Portugal, July 7-11, 1997, pp. 797-801.

[5] PRETZ, K. (n.d.). Building Smarter Wheelchairs: Making life a little easier for people who can't walk. *The Institute*. http://theinstitute.ieee.org/technology-focus/technology-topic/building-smarter-wheelchairs