The Pathfinder Zybot: an Open Platform Telepresence Robot

András Erdős[(1)], József Zákány[(1)], Stefan Oniga[(1),(2)]
University of Debrecen, Faculty of Informatics,
[(1)] Intelligent Embedded Systems Research Laboratory
[(2)] Department of Informatics Systems and Networks
Debrecen, Hungary
erdos.andris@gmail.com, zjozsi891018@gmail.com, oniga.istvan@inf.unideb.hu

***Abstract*— This paper presents an Open Platform Telepresence Robot developed by authors. The main goal was to develop an Open Platform for development of telepresence or assistive robots. The system has two main parts, a PC application for the users of the robot, and the robot itself. A friendly GUI was developed in order to control and track the machine. The robot is built using an iRobot Roomba as the base platform. The robot's main component is an FPGA based ZYBO development board that it is connected to a lot of peripherals. It runs an Ubuntu Linux operating system, which provides a solid base for the software. The PC application and the robot software are communicating with each other, using a unique protocol developed by authors. The robot also has a lot of useful features, such as remote controlled or autonomous movement, obstacle detection and avoidance, video streaming of the on board camera images, etc. Such a robot it is intended to be used for daily life assistance of older adults or persons with different type of disabilities.**

Keywords — Assistive-robot, navigation, mapping, autonomous, Zybo, Debrecen

I. INTRODUCTION

A. *Related works*

The increasing number of the elderly population along with increased costs related to assistance of their independent living at home leads to extreme challenges that could be solved only by using assistive technologies in general and assistive robots in particular. In this way elderly citizens would need less human assistance or they need human assistance at a much later stage in their aging process. The acceptance of technology still remains a delicate subject.

Major universities and research centers have research related to assistive and telepresence robots [1] - [13]. A special interest is related to Open Platform Telepresence Robots. For example there the TurtleBot is a low-cost, personal robot kit with open-source software. It has a Kobuki base, an ASUS Xion Pro Live, a Netbook, a Kinect, and a ROS system which can handle everything. Or another example is The Anybots QB telepresence robot. It can be operated from a web browser, the user can interact with other people through the robot as he would there. And we can mention the TILR of RoboDynamics. This robot is a video conferencing system on wheels. It can be controlled remotely and it provides a video stream for the user. There are many other robots, like Costai's Jazz Connect robot, The MantaroBot developed by Mantaro, and the VGo robot.

This research presents a work in progress, namely the development of an assistive assembly consisting of an assistive and telepresence robot platform together with the related components and services

We aim to develop a complex autonomous robot as a part of our assistive system for elderly local and remote assistance. The robot must pay close attention to the elderly person, try to learn his behavioral patterns, give him/her verbal advice, could control the room temperature, and could alert the family or doctor in case of the assisted person is not feeling well. An important role of the robot is to entertain, playing the favorite music, video or photo slideshow, or displaying favorite internet sites. It can respond to question regarding calendar (day, month and year), temperature indoor and outdoor, daily expected events and meetings. It should behave like a sensitive family member and react to the problems of the patient.

The main functions required for the assistive robot are:

- Manual control by keyboard or joystick
- Autonomous movement in patient's home
- User identification using an RFID tag
- Easy to use GUI for simple usage
- Video streaming, with the camera on the robot
- Dynamic obstacle detection
- Map builder application
- Shortest path determination

A good example of designing a telepresence robot can be found in [13]. This paper provides guidelines for designing a telepresence robot with features like video, audio stream, user interface and automated behaviors.

B. *Our project*

The project aims to design an Open Platform Telepresence Robot. The robots basic function is to help people in some ways. During the development we concentrated on functions like mapping, controlling, and navigating the robot and autonomous behavior. The project has two different parts. The first part is the robot including the hardware and the software, the second is the PC application. We can control the robot manually by keyboard or a joystick. For the autonomous movements, the robot needs a predefined map. The operator of the robot can upload a map (Fig. 2) of the room where he/she wants to use the robot in. It is a simple hand drawn image file, with black and white elements. The robot processes the picture and it creates a simple matrix from it, which will be used as a map. The user can use the GUI to put the robot on the map, and also determine a goal position. When the robot has the map and the goal position it will start moving. First, it uses an A* algorithm to assign the shortest route avoiding all the objects drawn on the map. All actions of the Zybot can be seen on the graphical user interface on the PC. We can follow the movements using the camera on the top of the robot. It is a simple video stream towards the user. If the robot got lost, we can use remote control function. Thanks to the RFID reader on the bottom of the robot, we can define reference points for orientation. If the robot finds an RFID card on the floor it can refresh its coordinates and it will know where exactly it is.

We used a Zybo as the central control unit of the robot. All the peripherals including the Roomba platform are connected to it. We also used a Wifly WiFi module and a unique protocol system to communicate with the PC.

II. PC SOFTWARE

A. Communication between a PC and the Robot

The data changing method between the Zybo's WiFi module and the PC is a simple UART communication. To describe the messages, we use a specific internal protocol system.

B. Graphical User Interface

During the development of the application, we focused on the user. Software included important features, which are the following:

- The application user interface is very easy to use.
- It has a graphical display surface that allows maps creation.
- It makes possible the management of the maps: save maps, load maps, modify maps.
- The application is expandable easily with new developments and functionalities. Because of this it includes general planning samples and models.
- The representation, the model, and the controller are totally independent parts.
- The application has a common protocol system with the robot.
- It provides an interface for the robot, through which it is able to insure two-way communication.

The graphical user interface (Fig. 1) is a windows-like surface, we have tiles for every function, for example sensor stream, video stream and logs. These are the information, which we acquire from the robot.

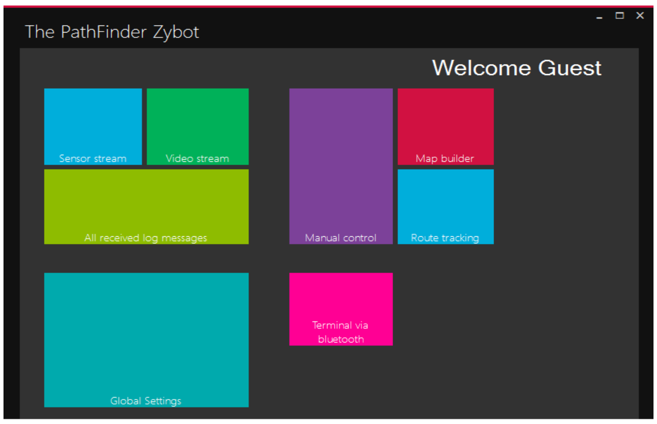

Fig. 1 – The Graphical User Interface of the PC application

There are tiles what we can use for controlling, driving the robot. The Manual control tile can be used as a remote control option. We can drive the robot from the keyboard of the PC or we can use a joystick.

The robot is able to act on its own. For the autonomous movements we have to use the Map builder and the Route tracking tiles.

C. Map builder

The map builder tile starts an application that helps to create maps. If we have a pre-made map or picture, we can just simple drag the picture and drop to the map builder window. We can draw extra objects on the map easily if we want to. We have placed several map editor tools in this window.

We can operate on the uploaded map by navigating to the route tracking tile. After that, we can put the robot on the map. If we are done, we can start the map upload to the robot.

The map builder (Fig. 2) slices the image to small rectangles (nodes) and the map becomes a grid. We can define the node size on the map and on the room. Those parameters will determine the resolution of the slicing. We can also define the size of the robot. The software will upload the map node by node to the robot.

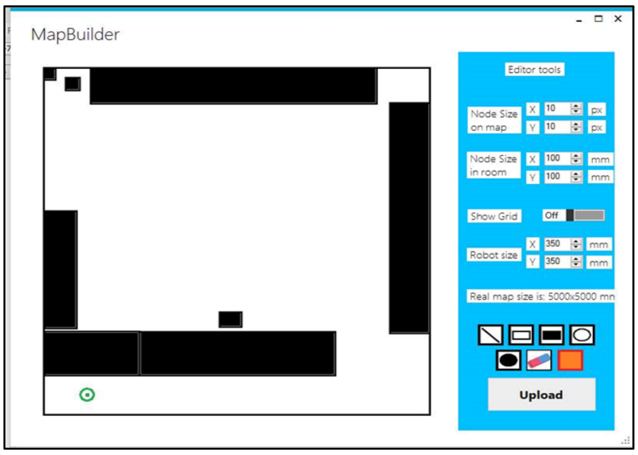

Fig. 2 - The Map Builder function of the PC application

D. *Route Tracking*

The size of the nodes is user defined. These parameters will determine the resolution of the grid. When the user uploads a map (which is an image), and set a goal point, the A* algorithm determines the shortest path. This process called the upper operation because the robot is not moving during the calculations. The lower operation is to try to navigate the robot to a specified location, and try to detect barriers, meanwhile it is moving.

Before The GUI sends out the image of the map, preprocess it. It slices the image into small nodes and send the nodes to the robot with the specified protocol.

The robot software collects all the information about the nodes (in which position, which node is empty) and the start position.

Calculating the robot's speed, and determining the robot's location is easier if we handle the robot as a material point of the grid. Our robot is circle based with 350 mm diameter. So the robot size is much bigger, than the node size. We can handle the robot, as a material point, if we fatten all the object on the map with the robot's radius. The method needs the following parameters: robot radius, node width, node height. First, we need to find the edges of the objects, after that we need to extend them. This will be our real map, which will be the world for the robot.

In the route tracking tile (Fig. 3) we can see what the robot see from the uploaded map. We can define a goal position for the robot - the red x.

The robot will determine the shortest route from its position to the goal and will send back the coordinates of the path. And finally, the robot will start moving by its own fallowing the path. To calculate the shortest route on this grid based map, we used an A* algorithm.

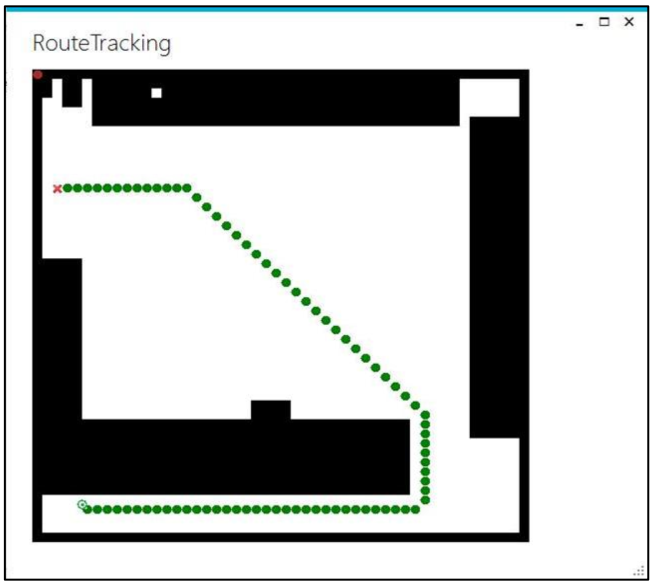

Fig. 3 - The Route Tracking function of the PC application

III. Hardware configuration

We have chosen an iRobot Roomba vacuum cleaner robot as the platform for our system. We have removed the parts we did not need and we have built a holder part for the system on the top.

The Roomba system contains two wheels driven directly by motors. It can be controlled by an UART interface with simple commands. Since we are not familiar with the mechanics, the Roomba system has proved to be a simple and good choice.

On this platform, we have built a system that can control and drive the robot. This system contains a CPU, several peripherals and the parts for the power supply. The central control unit is a Zybo board, on which we installed a Linux Ubuntu 15.04 operating system.

A. *Operating system*

In order to install the operating system, we have created two partitions on the SD card. One of them is the BOOT partition, which is a simple 100Mb FAT32 system. The other one is the ROOT_FS partition that is an ext4 file system and uses the rest of the space on the card.

The first file we made was the BOOT.bin. This file was created from three previously made files, the fsbl.elf (first stage boot loader) what we create with the SDK, the system_wrapper.bit which is the product of building the block design in the Vivado and finally the u-boot.elf what is the second stage bootloader, and we get it by compiling the u-boot system provided by Digilent.

The second file is the device tree, which contains the definitions of the peripherals. We had to add our modifications to the zynq-zybo.dts file and compile it to a .dtb (device tree blob) file.

The final part of the first partition is the uImage, which is technically the compiled kernel image.

On the second partition, we have to create a standard Linux file system. In our case, we used the Linaro file system. It can be downloaded from the Linaro's home page as a compressed file. We have to extract that file and synchronize all the files in it to the ROOT_FS partition of the SD card. (Fig. 4)

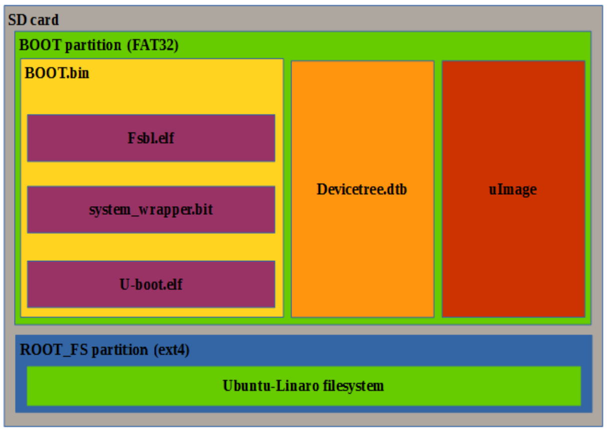

Fig. 4 - The architecture of the partitions and files on the SD card.

B. *Hardware on the FPGA*

We have created a block design (Fig. 5) for the ZYNQ processor that can operate with a Linux operating system.

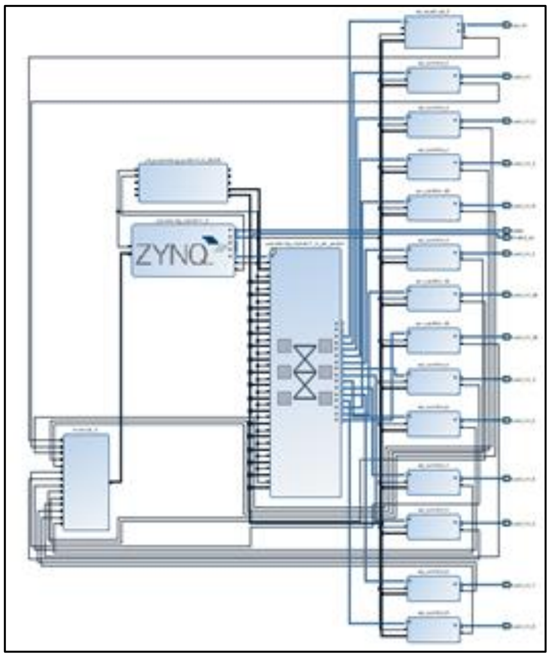

Fig. 5 - The Hardware architecture for the FPGA

In the design we used different peripherals. For example, we needed a lot of UARTs, so we created thirteen pieces of UartLites. The interrupts of the UARTs are connected with a logical OR module that is connected to the interrupt input pin of the ZYNQ. We also used the peripherals in the ARM part of the ZYNQ processor. UART1 is used as the serial terminal of the Linux, and the I2C-0 bus is controlling the magnetometer module. We had to enable these peripherals in the kernel menu config. For example, we had to find and enable the Uartlite, the I2C, and the ARM UART modules then compile the kernel with all of them.

IV. Robot software

A. *Software design*

The robot software is written in C++. We chose this language, because it is really fast, easy to write drivers and describe hardware close parts in it. The software itself is vertically divided, from the hardware layer to the artificial intelligent layer (Fig. 6).

The software describes strongly parallel operations. Each module has an own thread, and between the modules the communication take place via messages. The program essential features and the functionality are presented in bellow.

- If a sensor sends data, the Zybo is able to receive it, anytime.
- On the driver level, we can keep as many data as we need.

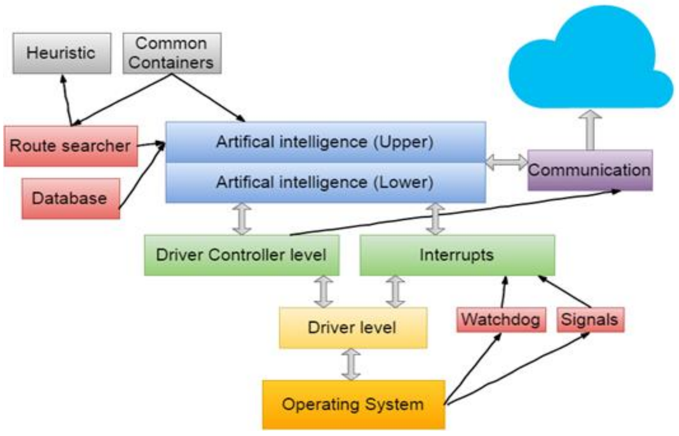

Fig. 6 - The layers and structure of the software of the Zybot

- If any higher level operation needs data, it is available anytime.
- It is easy to expand our system with new modules.

The threads describe a parallel functionality.

The driver level (Fig. 7) describes an abstract layer, through which our program interacts with the sensors. All the drivers have the same structure. This means, that if we want to add a new physical module to our project, we need to write a driver. We have a driver sample, so we can do it easily. When we create a new driver, we pass to it an object id and a port name. The port contains a serial port name (described by OS, example: ttyUL0), or an I2C bus address (example: /dev/i2c-0). The object id ensures, that every driver is unique and there is no other object that is identical with it. For example, if we want to handle 5 ultrasonic range finder, we can do it with one MaxSonar driver, and yet all the range finder drivers can be identified. The driver tries to open a socket on the specific channel. If it succeeds, the OS gives back a port id. With this ID, we are able to read/write across the socket. At the same time, the driver creates a thread. This thread periodically observes the communication channel. If a new data is available, the thread read it, and save it to a specific container. The driver provides interfaces to the outside through event handler functions.

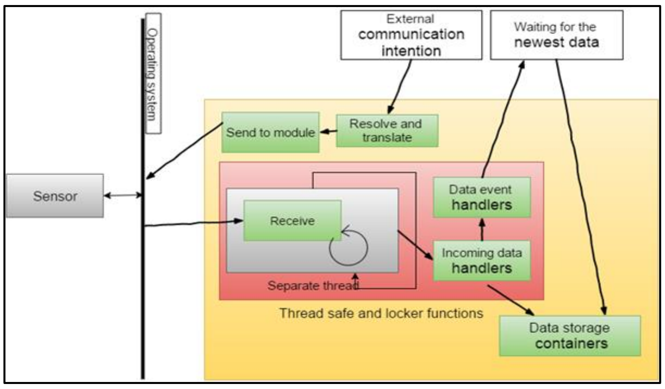

Fig. 7 - The Driver level of the software

B. *A* algorithm*

The A* algorithm is able to calculate the shortest path between two points (Fig. 8) on a map, with the given state-space representation. Our map is a grid-based map, and the robot center point is always at the center of a node of the grid. The robot can move node to node. To calculate the shortest route on this grid based map, we used A* algorithm.

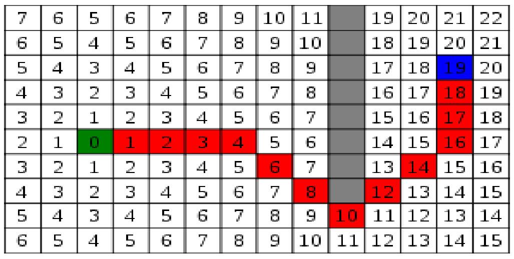

Fig. 8 – The shortest path determined by the A* algorithm

We determine a heuristic for every node. We tried out several types of them:

- Manhattan distance
- Diagonal distance
- Euclidean distance
- etc.

We count the Euclidean distance from the robot to the given node and from the node to the goal position. The heuristic is the sum of this two value (Fig. 9).

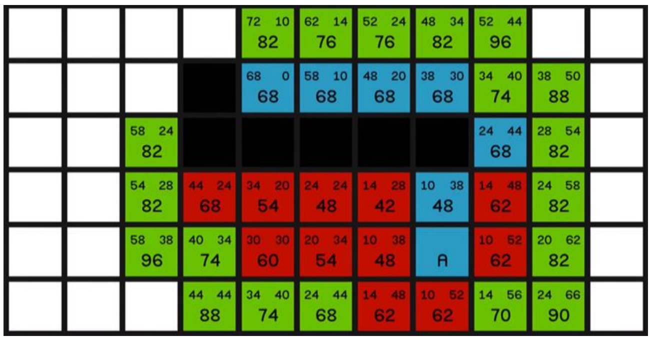

Fig. 9 – The calculations of the Heuristic

The A* algorithm has a set of opened and closed nodes. The closed nodes are already visited, the opened nodes are available to visit. The algorithm will always visit the node with the lowest heuristic value. Every node has a parent. It means that node from that we can reach the given node on the shortest path. When it finds the goal node, it will go back step by step to the start using the closed nodes' parents. It collects every node on the way back, and that will be the shortest path.

V. Conclusion

We achieved to develop an Open Platform for telepresence or assistive robots on FPGA based ZYBO development board. We have reached the first milestone for the development of a robot that is able to move on its own and navigate on a map which will be able to help people with special needs. The robot can be remotely controlled and is capable of autonomous movement, obstacle detection and avoidance.

ACKNOWLEDGEMENT

This work was made as a part of the ongoing research that takes place in the Intelligent Embedded Systems Research Laboratory at the University of Debrecen, Hungary, with the aims to assist elderly or sick people everyday independent activities using the latest assistive technologies.

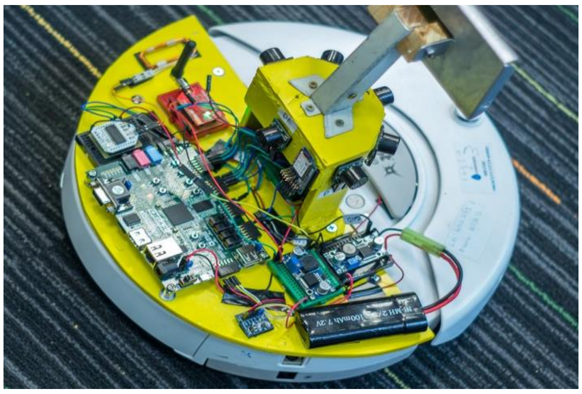

Fig 10 - The Pathfinder Zybot

The project (Fig 10) was presented at Digilent Design Contest Europe 2016, where won the third price.

References

[1] A. Lago, "DOMEO, Domestic Robot For Elderly Assistance. First," in AAL Forum, Lecce, Italy, 2011.

[2] V. Dupourqué, "DOMEO, an Open Robotic Platform for Cognitive and Physical Personalized Homecare Services," in Workshop PAL, Sophia Antipolis, France, 2011.

[3] E. Ackerman, "Suitable Technologies Introduces Beam Remote Presence System," IEEE Spectrum Robotics News, 26 September 2012.

[4] A. Mataric, A. Okamura and H. Christensen, "A Research Roadmap for Medical and Healthcare Robotics," Arlington, VA, 2008.

[5] A. Alexan, A. Osan and S. Oniga, "Personal assistant robot," in Proceedings of 2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging, Alba Iulia, Romania, 2012.

[6] A. Alexan, A. Osan and S. Oniga, "AssistMe robot, an assistance robotic platform," Carpathian Journal of Electronic and Computer Engineering, vol. 5, no. 1, pp. 1-4, 2012.

[7] I. Orha and S. Oniga, "Assistance and telepresence robots: a solution for elderly people," Carpathian Journal of Electronic and Computer Engineering, vol. 5, no. 1, pp. 87-90, 2012.

[8] J. Sütő and S. Oniga, "Remote controlled data collector robot," Carpathian Journal of Electronic and Computer Engineering, vol. 5, no. 1, pp. 117-120, 2012.

[9] D. Feil-Seifer and M. Mataric, "Human-robot interaction," in Encyclopedia of Complexity and System Science, new York, Springer, 2009, pp. 4643-4659.

[10] M. Heerink, B. Kröse, V. Evers and B. Wielinga, "Assessing Acceptance of Assistive Social Agent Technology by Older Adults: the Almere Model," International Journal of Social Robotics, vol. 2, no. 4, pp. 361-375, 2010.

[11] Ha M. Do, C. Mouser, W. Sheng "An Open Platform Telepresence Robot with Natural Human Interface" Proceedings of the 2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems May 26-29, 2013, Nanjing, China

[12] F. Michaud, P. Boissy, D. Labonte, H. Corriveau, A. Grant, M. Lauria, R. Cloutier, M.-A. Roux, D. Iannuzzi, M.-P. Royer : Telepresence Robot for Home Care Assistance

[13] M. Desai, K. M. Tsui, H. A. Yanco, and C. Uhlik: "Essential Features of Telepresence Robots Technologies for Practical Robot Applications" (TePRA), 2011 IEEE Conference on, pp. 15-20, 2011.