JOURNAL OF SCIENCE

Sakarya University
Journal of Science

VOLUME: 23
ISSUE: 4
AUGUST 2019

e-ISSN: 2147-835X

Sakarya University Journal of Science 23(4), 617-632, 2019

An Embedded System Design to Build Real-Time 2D Maps for Unknown Indoor Environments

Ercan Coşgun*[1], Hayriye Korkmaz[2], Kenan Toker[3]

Abstract

This paper presents a remotely controllable, differentially driven wheeled mobile robot development in order to build 2D maps for unknown indoor environments. This system would eliminate the need to pre-explore such environments. Main aim of the study is to develop a system with high accuracy by using minimum number of sensors and a processor with low cost especially for comparatively small indoor areas. The distance traveled was calculated using the wheel odometry method. Obstacles surrounding the robot, the distance traveled, and the robot's orientation were obtained using an ultrasonic distance sensor, optical encoder, and a 3D orientation sensor (also known as an Attitude and Heading Reference System –AHRS), respectively. In addition, the characteristics of the system hardware components were empirically explored, and the errors resulting from the sensors were evaluated. The non-linearity percentage error arising from the encoder was defined and then compensated for. The hysteresis behavior of the ultrasonic distance sensors was also empirically tested. All of the tasks were conducted by using a low-cost FPGA (Field Programmable Gate Arrays) board. A graphical development platform of National Instruments (NI) LabVIEW and its FPGA Module was preferred in the study for embedded system programming instead of the text-based HDLs (Hardware Description Languages). This distinguishes the proposed system from similar prior studies.

Keywords: Indoor Mapping, Wheel Odometry Errors, LabVIEW FPGA Module, distance measurement

1. Introductıon

Mapping is based on a set of measurements acquired from sensors supported by a plotting algorithm; it can therefore be defined as being a two dimensional modeling process of an environment. For a mapping system, the duration it takes to make a calculation, as well as its cost and accuracy, are significant criteria. As such, the sensors and algorithms used in the system directly affect the system's accuracy. The widespread use of global positioning systems (GPS) makes the mapping process easier for outdoor areas and allows for high-accuracy positioning. However, it is impossible to use GPS in indoor spaces, and so there are challenges in mapping and positioning processes for such areas [1, 2].

Localization is one of the biggest problems in robotics applications, because it is necessaryas to know the instantaneous position (x, y, Θ) of the robot with respect to its starting position (reference) to travel autonomously [3]. Once these data were obtained, localization is no longer a problem for the robot.

[1] Kırklareli University, Vocational College of Technical Sciences, Kırklareli, TURKEY. ORCID: 0000-0003-4387-3699
[2] Marmara University, Faculty of Technology Electrical-Electronics Engineering, İstanbul, TURKEY. ORCID: 0000-0002-5994-7587
[3] Marmara University, Faculty of Technology Electrical-Electronics Engineering, İstanbul, TURKEY. ORCID: 0000-0001-8568-4277

However, monitoring its motion subsequently becomes as a significant concern. More specifically, it is difficult to obtain detailed information about a robot's position if the starting point is unknown. In such cases, possibility based [4], Monte Carlo localization [5], and Kalman-filtered [3, 6, 7] systems are used.

One of the methods that can be used to determine the position of a robot in an indoor area is through the use of active beacons that have been placed in the environment beforehand. In this method, at least three beacons should be used. Signals received from these beacons are tracked and put into an algorithm, thereby giving position information. Active beacons may emit either light or sound. This method has been considered to be an indoor version of GPS [8].

Another method is to make use of landmarks in the area of interest. There are two types of landmarks that can be used: natural and artificial. In this method, the location of a mobile robot can be found through the placement of unique and distinguishable landmarks; this enables the robot to position itself according to the landmarks [8-12]. Natural landmarks are objects that already exist in an environment, and examples of such in indoor environments are doorjambs, walls, table legs, or edges. Landmarks can be a color that normally does not exist in the environment being mapped, or they may be a geometrical shape, such as a triangle or circle. Or they may contain modules including some digital data, such as a barcode or a RFID (Radio-frequency Identification). Since it is difficult to detect irregular geometrical objects; in the studies of [9], [10] and [13] artificial landmarks, such as RFID tags, were placed in the area of interest. The robot detects these objects, classify them, and determine its own position with respect to these landmarks. For example, in [12], an artificial landmark has been placed on a charging unit so that a mobile security robot could charge itself. When the robot's power level is too low, it finds that landmark in order to connect itself to the charging unit.

The disadvantage of the aforementioned methods is that they require a pre-exploration process. The system that we propose in this paper, however, does not require any pre-exploration.

One of the methods typically used to calculate the distance traveled by a mobile robot is wheel odometry (as it counts the number of pulses output by encoders). This method is important for indoor environments where GPS data is not accurate or available, such as inside buildings and tunnels, as well as underground. However, there are some problems with this method such as driving the wheels differentially, wheel slippage, and irregular floors. These can cause the robot to move in an undesirable direction and result in incorrect transformation processes between the Earth and a robot's coordinate frames [14-16]. If only this method is used in differentially driven wheeled robots, errors arising from the mapping system increase cumulatively.

Ojeda and Borenstein tried to reduce such errors by using three novel methods: 1) Fewest Pulses, 2) Cross-coupled Control, and 3) Expert Rules [17]. They implemented these methods on a modified Pioneer AT skid-steer platform and the results were point to clear advantages of the using third method over the other methods. Four independent drive motors with four encoders were used in the study and two identical cross-coupled PID (Proportional–Integral–Derivative) controllers were implemented for each pair of wheels on the left and right hand side. Another third controller was used to couple average pulses from left side to average pulses from the right side of the robot. Additionally, this multi-controller application was supported with two expert rules. By this means, the researchers sought to minimize the errors in the robot's odometry. Although having an encoder in every wheel was anticipated to increase the accuracy of the mapping system, it instead led to systematic and nonsystematic errors [7, 18].

Another study, attempted to address similar orientation and distance error problems for a two-wheel mobile robot using a PID controller [14].

An alternative solution for this problem was devised by Maimone et al. [19], as they used visual odometry in their study, with which they obtained a low error rate. However, it is not possible to use this method in every environment, because it necessitates the use of visual sensors that require a sufficient amount of light.

In order to address the errors arising due to wheel odometry, Yenilmez et al. used referenced landmarks [11]. The Newton recursive localization method was used to overcome these errors; in this method, 1 m lengths of 10 cm diameter cylindrical reference objects (whose positions were known) were used. However, there are two significant limitations with this method: the first is that a similar reference object should not exist in the environment that is to be mapped; the second is that pre-exploration is required. A certain reference object (whose position is known) has to be placed within the environment.

In encoder systems, however, it is impossible to find the distance traveled and the orientation of the robot if there is no contact with the ground. In the literature on odometry-free systems, inertial measurements are made using a combination of gyroscopes and accelerometers, and these are used to detect a robot's motion [20]. By using these measurements, the distance traveled and the orientation of underwater or flying robots can be readily determined.

Studies that do not require pre-exploration use optic encoders to find out the distance traveled by a robot. This allows for both the distance and orientation to be measured both digitally and in high resolution. In robotics applications, by using a single optical transceiver, the motor rotational speed and distance traveled can be calculated, but the orientation cannot. If the rotation direction is needed, two optical transceivers (i.e., a quad encoder) that measure data with a 90° phase difference should be used [21]. Another study found that the number of turns of the motors and the direction of the rotation of the wheels can be calculated by a quad encoder that uses Hall Effect sensors [22]. This allows for highly sensitive and reliable measurements to be made.

The work carried out in recent years on simultaneous localization and mapping (SLAM) has primarily focused on developing algorithms that are real-time implementable that can obtain more accurate and robust maps [23, 24. One of the ways to increase the accuracy of 2D map extractions in unknown environments is to develop sensor fusion applications [25] as considered in the system proposed in this study. It is also important that the position information be updated quickly, so that a robot in the field can determine its own position more accurately. In addition, the processing speed of the platform on which the algorithms is run is also very important. Because in order to create a meaningful map, there must be a platform to deal with the computational load and complexity required in data processing [26].

While CPUs are sequential processing devices, FPGAs are parallel processing devices. FPGAs can outperform CPUs in executing certain tasks, because FPGAs can do multi-loop controls at different rates by using the Single-Cycle Timed Loop (SCTL) thanks to LabVIEW module [27]. This kind of loops execute functions inside within one tick of the FPGA clock (in this case, one tick is equal to 1/50 MHz = 20ns). As a result, LabVIEW FPGA module was preferred for use in this study due to its ability to perform true parallel processing. They are also able to respond to all of the timing constraints (using extremely fast loop rates) mentioned above.

We developed a sensor fusion application that used a high-speed FPGA of 50 MHz. FPGAs became programmable in many different ways with the increasing popularity of embedded systems. Many languages and programing platforms (such as LabVIEW FPGA tool or HDL Coder from Matlab) are now available with new extensions to the existing tools. They can be used as an alternative way to the most common HDLs (i.e., Verilog and VHDL). One of these is the FPGA Module add-in, which the NI LabVIEW graphical design platform offers. The most important feature of NI LabVIEW, apart from being able to use other programing languages in it, is that it offers the ability for different loops to operate in parallel much better than other platforms do [27, 28].

In this sense, the work presented by Gong et al. is the most similar to the solution proposed in this paper. In this study, a faster embedded system JetSon TK1 from NVIDIA Corporation was preferred for use instead of an ordinary CPU. However, they used an additional interface card Arduino Mega2460 microcontroller board for their data acquisition purposes, and the motor driving/control application. This prevents the embedded system from fully exploiting the advantages (in terms of reaching fast loop execution rates) offered by it [29].

2. METHODOLOGY

In this study, we designed and used a remotely controllable differentially driven two-wheel mobile robot, called MappingBot; its architecture is shown in Fig. 1. User interaction and visualization lived on the PC, while motor control, PID controller for odometry errors and critical high-speed processing tasks including multi-sensor data acquisition and timing tasks lived on the FPGA (See Appendix A for MappingBot Movement Control detailed block diagrams).The main contribution of the paper is to setup an accurate measurement system in order to draw indoor maps of the interested region without the need for any pre-exploration processes; in order to do this; we have decided to use a low-cost FPGA board as well as avoiding using expensive sensors, such as LIDAR (Light Detection and Ranging). All of the embedded (data acquisition, PID controller for odometry errors, and motor driver) and remote operations (see Fig. 1), including wireless communication between PC and the robot, were developed in a platform that enables graphical embedded programming.

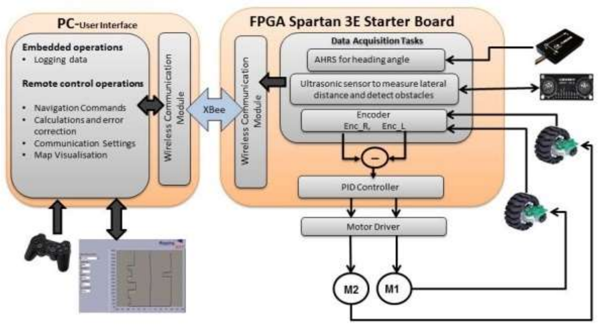
PC-User Interface
Embedded operations
• Logging data
Remote control operations
• Navigation Commands
• Calculations and error correction
• Communication Settings
• Map Visualisation
Wireless Communication Module
XBee
FPGA Spartan 3E Starter Board
Wireless Communication Module
Data Acquisition Tasks
AHRS for heading angle
Ultrasonic sensor to measure lateral distance and detect obstacles
Encoder
Enc_R, Enc_L
–
PID Controller
Motor Driver
M2
M1

Figure 1. MappingBot Hardware Architecture

The proposed system uses the wheel odometry method and is supported by a PID controller and an AHRS sensor to minimize any odometry errors. In our study which uses a common and simple method, it is sufficient to sense the distance between the surrounding objects and robot. When the studies requiring no pre-exploration reviewed, it can be seen that some sensors, such as gyroscopes, accelerometers, AHRS sensors, magnetic and digital compasses, encoders, LIDAR, ultrasonic sensors, infrared distance sensors were used [29-34]. In addition, those sensor data was werostly combined with camera data (RGB-D and/or thermal) as used in Google's Cartographer and Hector [35]. Combining sensory data derived from disparate sources is more difficult than technology using landmarks or RFID tags. For instance, in the odometry method, if the wheels drift or there is angular displacement, robot position will be calculated incorrectly. If no feedback is received while the robot is travelling (or moving), speed is considered to be constant. The position calculated through this method could likely be wrong, because the motor may decelerate/accelerate due to spinning or slippery ground. Therefore, the actual velocity data should be determined through the feedback received from the motors, so that a true and valid position can be calculated.

Therefore, in this proposed study, we have used a single PID controller to ensure that both of the differentially driven motors rotate at the same speed without an increase in the number of encoders (as given in Appendix A) which is as Szöcs et al. did [36]. And we supported the odometry data by an angle sensor. Thus, we endeavored to reduce the wheel odometry errors resulting from both the system and friction to a minimum [17, 36].

As mentioned earlier, by using a relatively new platform (NI LabVIEW FPGA Module) for the software development part of the MappingBot brings the study a true novelty. Although the use of an embedded system, such as a FPGA, is quite common in commercial applications, its use is relatively infrequent in the field of control and automation [37-41]. Embedded design has recently become more complex, with the size of the code required nearly decoupled over the last five years; meanwhile, the number of non-embedded experts who need to use such technology has also increased [42]. In addition, there is a consecutive transaction that needs to be carried out when using application development platforms provided by leading companies such as Vivado or ISE Design Suite by Xilinx Inc. As a result, there is a strong need for a new approach to embedded system designs in order for engineers who do not have sufficient digital design skills to work on these applications. These limitations have been addressed in by next-generation approaches, such as the NI LabVIEW FPGA Module in graphical embedded design. In this way, this module allows designers to use only one platform for every stage of an embedded design [38-42]. Due to NI LabVIEW has been reduced the workloads required for writing code in text-based programing languages, such as HDLs, development time was shortened. [42].

3. Hardware

In this section, the entirety of the sensing instrumentation of the MappingBot is listed and explained in detail.

3.1. Quadrature Encoder

A quadrature encoder, working with a 42 × 19 mm wheel, was used to find out the distance the robot traveled [14, 17, 18, 22]. Two infrared reflectance sensors were placed in the encoders. These were spaced so as to provide waveforms that were approximately 90° out of phase, which allow to determine the direction of rotation and provide four counts per tooth by a resolution of 48 counts per wheel rotation [43]. The location information, as well as the movement direction (forward and backward), can be obtained from two digital outputs that are provided by each encoder. The encoder is pre-calibrated for 5.0 V operations, but it can be modified so as to be used at 3.3 V for FPGA applications, as has previously been described [43].

The encoder produces 48 pulses per wheel cycle. In order to find the distance required to produce one pulse, the circumference (C) of the wheel was divided by 48, as demonstrated in (1):

$$\frac{C}{48} = \frac{2x3.14x\left(\frac{42}{2}\right)}{48} = 0.274\, cm/pulse \qquad (1)$$

For example, if the desired distance is equal to 2 cm, then the number of pulses to be counted can be calculated by using (2):

$$\begin{matrix}Number \\ of\ pulses\end{matrix} = \frac{2\, cm}{0.274\frac{cm}{pulse}} = 7.29\, pulses \qquad (2)$$

However, the closest higher integer, 8, was preferred in practice. Thus, when 8 pulses were counted, the distance traveled could be calculated as in (3):

$$\begin{matrix}Travelled \\ Distance\end{matrix} = 0.274\frac{cm}{pulse} \times 8\, pulse = 2.192cm \quad (3)$$

During the pre-experiments that we conducted, the distance travelled along the y-axis were measured by setting the encoder resolution to 1.096 cm (or 4 pulses), 2.192 cm (or 8 pulses), or 3.288 (12 pulses) cm, and the results were then compared with the reference length of 400 cm. The percentage error was calculated using the absolute change between the experimental (measured) and theoretical (actual) values, and by dividing the sum by the theoretical (actual) value, as shown in (4):

$$\%error = \frac{\underset{(measured)}{Distance} - \underset{(actual)}{Distance}}{\underset{(actual)}{Distance}} \qquad (4)$$

As seen in Table 1, the lowest percentage error was achieved using the 2.192 cm resolution.

Table 1: Percentage errors for different resolutions

Resolution(cm)	Measured distance(cm)	% Error
1.096	403.287	0.82
2.192	401.238	0.38
3.288	406.696	1.48

To obtain meaningful displacement information from the encoder, a part of algorithm given in NI Quadrature Encoder Example [44] was integrated into our application (see Appendix B for block diagram of the encoder subVI). As is outlined in the program flowchart in Fig. 2, the "Receive Data" Boolean control becomes active (true) when every 2.192 cm displacement has been measured; all the sensor data is then acquired and transmitted wirelessly to the remote interface.

3.2. Ultrasonic Distance Sensor

The ultrasonic distance sensor (URM 37) from DFRobot has three operating modes, and these modes can be changed by writing 0x00, 0x01, or 0x02 to the EEPROM through the serial port [45]. In this study, Mode 1 (passive serial control mode) was chosen so that the sensor waits for a command to measure and sends data instead of receiving data continuously; this allows the system to operate more efficiently. To begin with, a 4 byte data, with a decimal code of 34-0-0-34, is sent to the sensor in order to prepare it for the measurement task (for details, see Appendix C). To complete the transmission task, the "for loop" must run four times. This is a waiting duration of 104 µs and is calculated by using (5):

$$Waiting\ Time \left(\frac{1}{\begin{matrix}Baud \\ Rate\end{matrix}}\right) = \frac{1}{9600} = 104\ \mu s \qquad (5)$$

That duration and its multiples were added into the flat sequence structure in LabVIEW block diagrams in order to send/receive data whose frame structure is given in Table 2. (For details, see Appendix C).

Table 2. Structure of the data frame of the ultrasonic distance sensor

	Wake-up	Start	Data	Stop
Number of bits	2 bits	1 bit	8 bits	1 bit
Duration	208 µs	104µs	832µs	104µs

When the reading process is complete, a data packet, whose structure can be seen in Table 3, is transferred to the URM1 array. Because the restricted area used in this application does not exceed 255 cm, D_H is always "0b00000000".

Table 3. Structure of Read Data

Command	D_H	D_L	SUM (Control Byte)
22(hex)	Distance Data-high	Distance Data-low	(command+ D_H+D_L)

3.3. 3D Orientation Sensor

The FY-AHRS-2000B is an ultra-miniature orientation sensor (also known as an AHRS) from Guilin Feiyu Technology Incorporated Company [46]. It consists of gyroscopes, accelerometers and magnetometers on all three axes and an on-board processor. It works with the serial communication protocol.

In order to use an AHRS the following steps must first be conducted in order; serial communication configuration, initialization, and magnetic calibration. After the required settings (baud rate = 9600) are completed using the supplied middleware, the initialization process is conducted. According to the user manual [46], if any of the axial rates (X rate, Y rate, Z rate) goes beyond ±1° when the AHRS module is in the static state, the gyro must be reinitialized. The experiments were carried out using the following gyro values: X rate = 0.18, Y rate = −0.17, Z rate = −0.71. This initialization process was performed only once before the sensor was mounted onto the robot, and it was assumed that these values would remain constant during the test. After the initialization was successfully completed, the calibration was carried out in order to eliminate the hard and soft iron distortions caused by the magnetic fields. After all of these tasks were completed, the initial parameters were written in FLASH memory in order for them to be permanently retained. The only limitation when using the sensor is that the rotation rate should not be more than 200° per second; the recommended rate is 100° per second or less [46]. During the experiments conducted, the rotation rate was far below this value. An array of 10x1 bytes was built using the data acquired from the AHRS, as seen in Table 4. In this study, only a 6-byte heading angle data (Field 3) was used, and the rest of the data was not subjected to any process (see Table 4 and Appendix D). The TX pin of the AHRS was wired to a digital input terminal of the FPGA board, and it was also wired to a Boolean control on the block diagram through the LabVIEW FPGA Module. When the Boolean control was set to “true,” a remote control module sends data to robot; when it switches to “false,” the module performs a data receiving operation.

The angle value can take any value between 0–360°. For example, if the value is 150.05°, then the data format is expressed, as shown in Table 4. Only the integer part of the angle value was taken into account in calculations. However, the instant angle value can sometimes have three digits, while at other times it may have two. Therefore, it was not possible to distinguish the integer part of the angle value only by entering a fixed number of bytes. In order to solve this problem, a desired value was obtained by searching for the comma (which is actually 44, or the corresponding ASCII character for a comma) in the data frame, as seen in Fig. 3 and Appendix D.

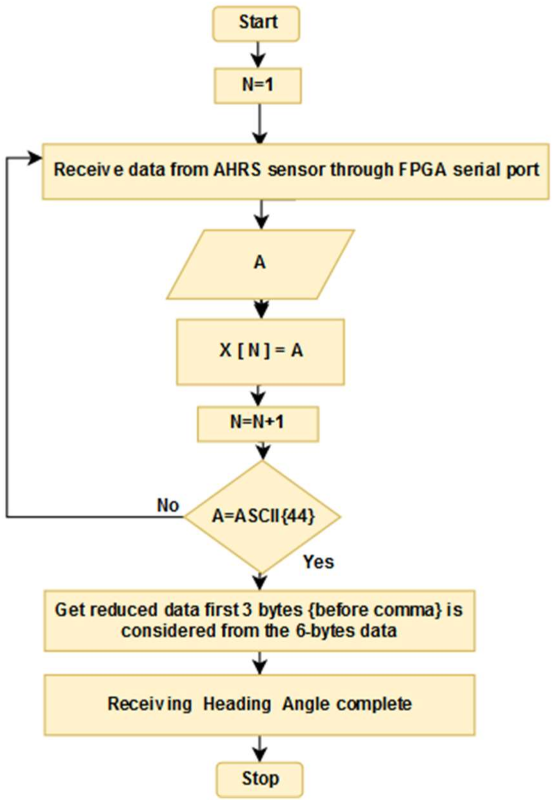
Start
N=1
Receive data from AHRS sensor through FPGA serial port
A
X [N] = A
N=N+1
No
A=ASCII{44}
Yes
Get reduced data first 3 bytes {before comma} is considered from the 6-bytes data
Receiving Heading Angle complete
Stop

Figure 3. Procedures for the heading angle measurement by the orientation sensor

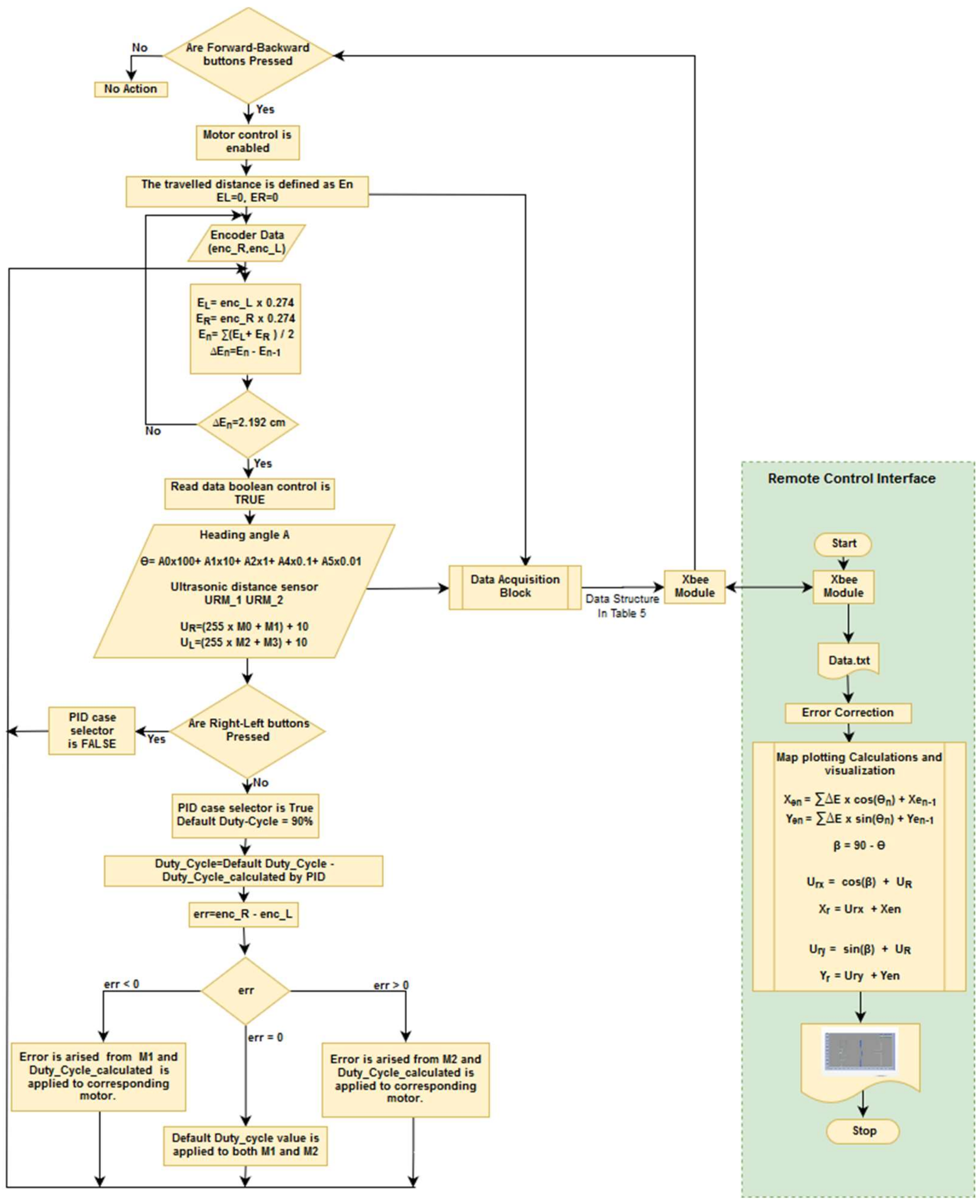
Are Forward-Backward buttons Pressed
No
No Action
Yes
Motor control is enabled
The travelled distance is defined as En EL=0, ER=0
Encoder Data (enc_R,enc_L)
E_L= enc_L x 0.274
E_R= enc_R x 0.274
E_n= ∑(E_L+ E_R) / 2
ΔE_n=E_n - E_{n-1}
ΔE_n=2.192 cm
No
Yes
Read data boolean control is TRUE
Heading angle A
Θ= A0x100+ A1x10+ A2x1+ A4x0.1+ A5x0.01
Ultrasonic distance sensor URM_1 URM_2
U_R=(255 x M0 + M1) + 10
U_L=(255 x M2 + M3) + 10
Data Acquisition Block
Data Structure In Table 5
Xbee Module
Are Right-Left buttons Pressed
Yes
PID case selector is FALSE
No
PID case selector is True Default Duty-Cycle = 90%
Duty_Cycle=Default Duty_Cycle - Duty_Cycle_calculated by PID
err=enc_R - enc_L
err
err < 0
err > 0
err = 0
Error is arised from M1 and Duty_Cycle_calculated is applied to corresponding motor.
Error is arised from M2 and Duty_Cycle_calculated is applied to corresponding motor.
Default Duty_cycle value is applied to both M1 and M2
Remote Control Interface
Start
Xbee Module
Data.txt
Error Correction
Map plotting Calculations and visualization
X_{en} = ∑ΔE x cos($Θ_n$) + Xe_{n-1}
Y_{en} = ∑ΔE x sin($Θ_n$) + Ye_{n-1}
β = 90 - Θ
U_{rx} = cos(β) + U_R
X_r = Urx + Xen
U_{ry} = sin(β) + U_R
Y_r = Ury + Yen
Stop

Figure 2. Flowchart of the entire system

Table 4. Data format of the orientation sensor

Field1	**Field2**	**Field 3**						**Field4**	**Field5**	**Field6**	**Field7**	**Field8**	**Field9**	**Field10**
Pitch	Roll	Heading						X	Y	Z	X	Y	Z	Resultant Acceleration
Angle		Angle						Angular Rates			Acceleration			
		A0 1	A1 5	A2 0	A3 .	A4 0	A5 5							

4. SOFTWARE (USER INTERFACE)

The flowchart of the proposed system was given earlier in the text (see Fig. 2). As can be seen in Fig. 4, the software is composed of two different interfaces; an embedded operations interface (Fig. 4a) running on the FPGA, and a remotely controllable operations interface (Fig. 4b) running on a PC. These were developed using NI LabVIEW 2015 in order to send navigation commands to the robot, to acquire and process all of the data provided by the sensor, and to visualize the map.

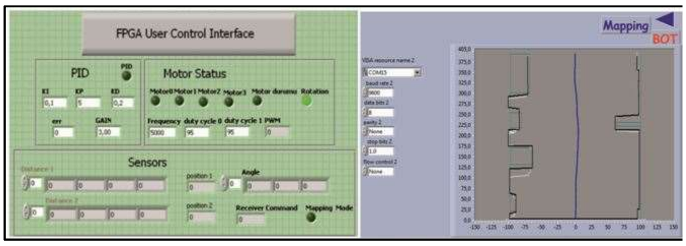

Figure 4. User interfaces for the MappingBot: a) Embedded operation's interface, b) Remote control operation's interface

In addition to using the ready-to-use VIs of NI LabVIEW during our system's development, some specialized sub-VIs, which had been customized according to the configurations and settings of the sensors and actuators, were also designed so as to be used in the embedded operations. These sub-VIs were as follows: 1) Serial Communication Read and Write Sub-VIs (were developed in order to interface between the FPGA and all sensors supporting the serial communication protocol. Here, both RX and TX tasks of ultrasonic distance sensor and wireless comm. module and also RX task of 3D orientation sensor were performed.) 2) PID Sub-VI—used for detecting and reducing wheel odometry errors (here the NI Position Estimation solution [44] was integrated into our own applications and details can be seen in Appendix E) and 3) Encoder Sub-VI—used for counting pulses and calculating the distance traveled in centimeters (given in Appendix B). This Sub-VIs could also be adapted so as to fit the work of any designer who used either the same, or similar, hardware. As such, users are able to easily incorporate into the system different sensors that they may need in future. They can also easily develop applications for robots that could serve different aims to ours.

Once the data acquisition was complete, the data obtained from the Xbee module, including all of the data provided by the sensors, was subjected to mapping calculations. In Table 5, the 12-byte data format and relevant labels are presented (see Appendix F for detailed view of XBee_RX and XBee _TX SubVIs' block diagrams).

Table 5. Data structure and labels for wireless communication

Right Ultrasonic sensor URM_1	Left Ultrasonic sensor URM_2	AHRS	Right Encoder	Left Encoder
M0-M1	M2-M3	A0-A5	encR	enc_L

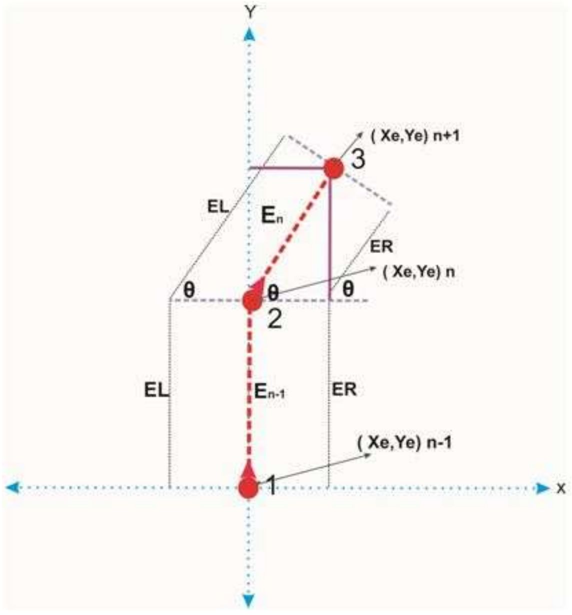

Figure 5. Finding the instantaneous position of the robot on the x–y axis

4.1. Odometry Localization

The limitations on the area where the MappingBot can be used in as follows: the distance on the x-axis is limited to 8 meters, depending on the sensors used; there is no limitation on the y-axis, so this distance can be increased as much as desired. The constraint on the y-axis is the range of the wireless communication unit that enables communication between the robot and the remote computer. The indoor distance is about 50 meters when the ZigBee technology is used, depending on the thickness of the walls. It is possible to increase this distance by using a GSM module; nevertheless, we chose to use an Xbee module (XB24-B) from Digi in this study [47].

There is one ultrasonic distance sensor on the left- and one right-hand outer edges of the robot. As can be seen in Fig. 5, while the robot makes a straight motion between points 1 and 2; it has moved with an angle between points 2 and 3.Thus, the sensors follow the robot's motion and rotate at the same angle. In order to calculate the actual distance, the distance between the midpoint of the robot and the outer edges should be added on the distance measured by the sensors. Since the width of the robot is 20 cm, half of this amount is added to sensed data in order to calculate the true distance between the obstacle and the robot, as shown in (6) and (7):

$$U_R = (255 \times M0 + M1) + 10 \qquad (6)$$

$$U_L = (255 \times M2 + M3) + 10 \qquad (7)$$

The distance traveled between two samples is calculated by counting the pulses from the encoder (See (1), (2) and (3)) and multiplying by a constant value of 0.274, as given in (8) and (9):

$$E_{Rn} = enc\_R \times 0.274 \qquad (8)$$

$$E_{Ln} = enc\_L \times 0.274 \qquad (9)$$

Instead of considering the pulses counted from only one encoder (even they are derived identically), the average pulse is calculated by using the right- and left-hand encoders in order to obtain a more accurate map.

The ϴ angle value, obtained from the orientation sensor, is calculated by using the following equation:

$$\Theta = A0 \times 100 + A1 \times 10 + \ldots$$

$$\ldots + A2 \times 1 + A4 \times 0.1 + A5 \times 0.01 \qquad (10)$$

The output of the orientation sensor gives the angle value of the robot's instantaneous position with respect to the north (North Magnetic Pole, NMP). However, the robot's motion direction cannot always be congruent with the NMP. Therefore, in this study, the initial value was taken as a reference in order to make the necessary calculation. To find out the true angle, first, the difference between two samples was calculated, and then it was added to 90° by considering the sign of the angle; a positive sign means that the MappingBot rotates clockwise (CW), while a negative sign means that the MappingBot rotates counter-clockwise (CCW).

In order to determine the angular displacement, the difference between the previous and current position of the robot was calculated. If the robot travels straight forward, the angular difference is 0°; if not, this means the robot has a rotational motion. In this case, the distance between the robot and the right/left obstacles (UR, UL) with respect to (0,0) was calculated based on the formula and definitions given in Table 6.

5. EXPERIMENTS

A restricted reference area (200 × 400 cm) was created in the laboratory, as seen in Fig. 6. Several square or rectangular shaped objects, whose positions and dimensions were known, were then placed. The mobile robot was subsequently set so as to navigate this reference area in order to acquire data for the mapping system. The map drawn according to the results of the measurement were compared with the reference map, and errors obtained from the different sensors were examined.

The experiments were designed to test the static/dynamic and statistical characteristics of the system such as linearity, hysteresis, and repeatability.

5.1. Experiment 1: Measurement of Distance Traveled on the Y-axis (Encoder Only)

A number of measurements were made in order to examine the characteristics of the system. The encoder resolution was set as 8 pulses, which meant that the data was sampled at 2.192 cm intervals.

In order to test the repeatability of the system, the distance of 400 cm, from point A to B, was repeated five times in different times and was recorded to Table 7. The percentage error was calculated using (4). We considered the distance (measured) as being the average of the five trials; the actual distances are given in the first column of Table 7.

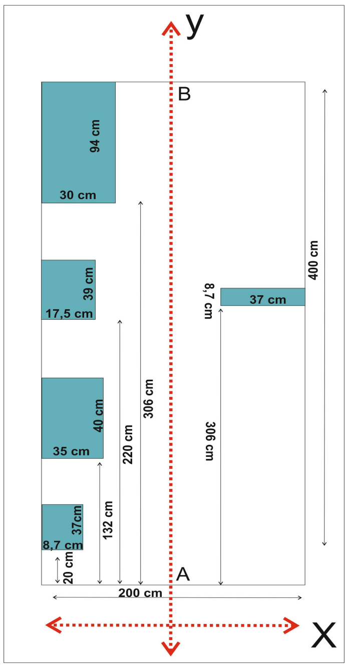

Figure 6. Restricted reference area

Table 6. List of Abbreviation

Parameters	Description	Formula
U_R	Actual distance between robot and right obstacles by taking mid- point as a reference.	$U_R = (255 \times \mathrm{M0} + \mathrm{M1}) + 10$
U_L	Actual distance between robot and left obstacles by taking mid- point as a reference.	$U_L = (255 \times \mathrm{M2} + \mathrm{M3}) + 10$
ϴ	Angle value obtained from AHRS.	$\Theta = A0\,x100 + A1\,x10 + A2\,x1 + A4\,x0.1A5\,x0.01$
β	Angle value between Ultrasonic1-Ultrasonic2 lines and X-axis.	(90- ϴ)
ERn	Calculated distance between two samples by using acquired data from the right-side encoder.	$E_{Rn} = \mathrm{enc\_R} \times 0.274$
ELn	Calculated distance between two samples by using acquired data from the left-side encoder.	$E_{Ln} = \mathrm{enc\_L} \times 0.274$
enc_R	Instant pulse value obtained from right-side encoder.	Directly acquired from the sensor.
enc_L	Instant pulse value obtained from left-side encoder.	Directly acquired from the sensor.
Xe_n	Corresponding X-axis value for instant robot position.	$\mathrm{Xe_n} = \sum_{n=0}^{n} \Delta\mathrm{E} \times \cos(\theta_n) + \mathrm{Xe_{n-1}}$
Ye_n	Corresponding Y-axis value for instant robot position.	$\mathrm{Ye_n} = \sum_{n=0}^{n} \Delta\mathrm{E} \times \sin(\theta_n) + \mathrm{Ye_{n-1}}$
Xe_{n-1}	Corresponding X-axis value for previous robot position.	-
Ye_{n-1}	Corresponding Y-axis value for previous robot position.	-
E_n	The amount of robot's midpoint displacement along y-axis.	$\mathrm{E_n} = \sum_{0}^{n} \frac{\mathrm{E_{Rn}} + \mathrm{E_{Ln}}}{2}$

Table 6. List of Abbreviation (Continuous)

Parameters	Description	Formula
ΔE	The amount of robot's midpoint displacement between the current and previous samples.	$\Delta\mathrm{E} = \mathrm{E}_n - \mathrm{E}_{(n-1)}$
Urx	Instant distance between robot midpoint and right-side obstacles along X-axis.	$\mathrm{U}_{\mathrm{rx}} = \cos(\beta) \times \mathrm{U}_{\mathrm{R}}$
Xr	X component value of distance between right obstacle and referenced X-Y plane.	$\mathrm{X}_{\mathrm{r}} = \mathrm{U}_{\mathrm{rx}} + \mathrm{Xe}_{\mathrm{n}}$
Ury	Instant distance between robot midpoint and right-side obstacles along Y-axis.	$U_{ry} = sin(\beta) \times U_R$
Yr	Distance value between right obstacle and robot on Y-axis.	$Y_r = U_{ry} + Ye_n$

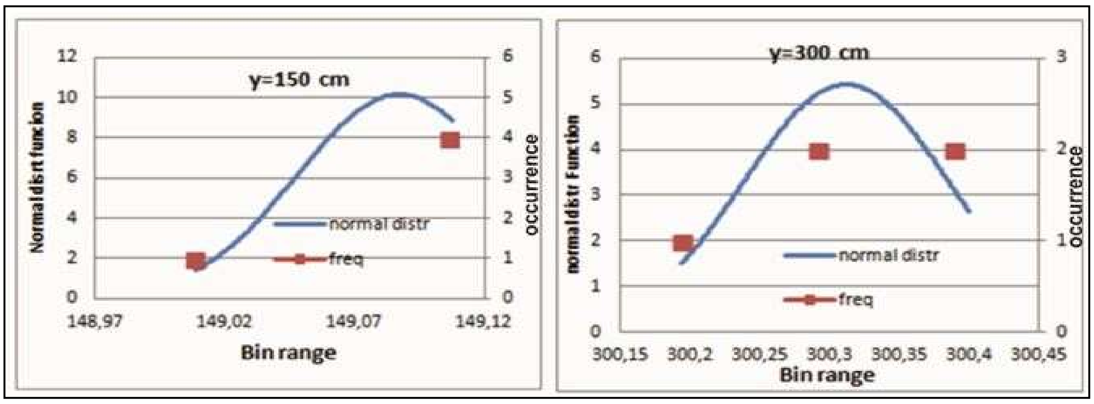

Figure 7. Repeatability experiment results and evaluation a) for 150 cm and b) 300 cm

Table 7. Repeatability experiment results

Actual traveled distance Along the y-axis (cm)	Trial (measured raw data)					Average (cm)	% error
	1	2	3	4	5		
50.416	51.3	51.3	51.3	51.3	51.3	51.3	1.753
100.832	102.6	102.6	102.6	102.6	102.6	102.6	1.753
149.056	151.7	151.6	151.7	151.7	151.7	151.68	1.760
199.472	202.9	202.8	203	202.9	202.9	202.9	1.719
249.888	254.2	254.1	254.3	254.2	254.2	254.2	1.726
300.304	305.6	305.4	305.6	305.5	305.5	305.52	1.737
350.72	356.9	356.7	356.9	356.8	356.8	356.82	1.739

Table 8. Repeatability experiment results after the correction was applied

Actual traveled distance Along the y-axis (cm)	Trial (after error correction) 1	2	3	4	5	Average (cm)	% error	StdDev
50.41	50.41	50.41	50.41	50.41	50.41	50.414	0.004	0
100.83	100.84	100.84	100.84	100.84	100.84	100.84	0.010	0
149.05	149.10	149.00	149.10	149.10	149.10	149.09	0.021	0.0393
199.47	199.43	199.33	199.5	199.43	199.43	199.44	0.018	0.0622
249.88	249.86	249.76	249.96	249.86	249.86	249.86	0.009	0.0622
300.30	300.39	300.19	300.39	300.29	300.29	300.31	0.003	0.0736
350.72	350.81	350.62	350.81	350.72	350.72	350.74	0.06	0.073

To begin with, an x–y graph was plotted using the actual and measured distance values. If the ideal and measured curves did not match one another, this meant that there was an error. A linear curve fitting function was then performed in order to define the error correction procedures, if any gain or offset errors existed. After the regression analysis, a positive gain error of 1.74% (the average of the observed individual errors for the five trials) was obtained. Finally, a simple mathematical correction technique was used to reduce the errors by assuming that the ideal transfer function is a straight line. After the necessary corrections were made on the raw data, the error resulting from the encoder sensor has been reduced until 0.01% for some cases, as shown in Table 8.

The distribution curves and the histograms showing the system repeatability are presented in Fig. 7. The results from the experiment show that, as the distance traveled increased, the standard deviation of the data obtained from the system also increased. This result points out an existing issue with the wheel odometry method. In Figure 7, Bin range stands for the full range of variation (all different measured values from minimum to maximum).

5.2. Experiment 2: Visualization of the Reference Area Map (Ultrasonic Sensor + Encoder)

In this experiment, the resolution of the encoder was again set to 2.192 cm, and objects were placed at either side of the referenced area; the positions of the objects, the distance between these objects, and the dimensions of the objects were then measured. These measurements were also repeated five times, and they were recorded in Table 9. In these measurements, the correction was applied to the distance traveled on the y-axis.

Table 9. Repeatability experiment results and evaluation of the distance measured between objects and the robot along the x-axis

Actual distance between objects and robot along x direction (cm)	Trial (measured raw data from left ultrasonic sensor in cm) 1	2	3	4	5	Mean (cm)	Error (cm)	% Error
100	96	93	96	95	94	94.8	5.2	5.20
91.3	92	90	93	91	91	91.4	0.1	0.11
100	102	98	103	101	99	101	0.6	0.60
65	65	61	62	67	64	63.8	1.2	1.85
100	101	96	101	100	98	99.2	0.8	0.80
82.5	83	77	83	83	79	81	1.5	1.82
70	76	69	72	78	72	73.4	3.4	4.86

By combining the data acquired from both the ultrasonic sensors and the encoder, the plotted map and the referenced area could be compared on the same scale, as in Fig. 8.

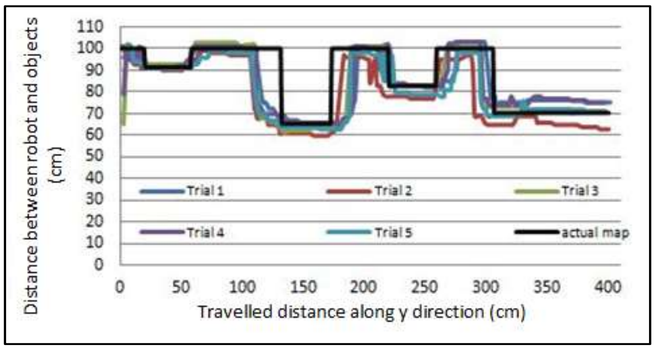

Figure 8. Comparison of plotted maps from each trials and actual map (only left side)

As can be seen from Fig. 8, the measured sizes of the objects were larger than the actual sizes. The detection range of the ultrasonic distance sensor was between 4–5000 cm, and its resolution, as seen from the data sheet, is 1cm [43]. However, the fractional digits shown in Table 9 indicate that they were a result of the calculations, and that the robot had an angular displacement. Another error resulted from reflected signals coming from the intersecting lines between the objects and the floor of the room. Errors in the positions where a change in distance occurs were higher than in the positions where the distance stayed constant. For example, the error in the area corresponding to the range of 50 to 100 cm was 0.35%. In order to minimize such errors, expensive LIDAR sensors can be used instead of the ultrasonic distance sensors.

5.3. Experiment 3: Hysteresis Evaluation

In this experiment, the starting point of the robot was defined as A, and the robot is let to travel toward point B. The starting point was then changed as B, and the robot start to travel on the opposite direction (from B to A). Only one measurement was taken for each case. Changing the robot's movement direction was not expected to cause any change on the measured distance. However, the maximum hysteresis percentage which is calculated by using (11) was observed as being 10% with 0–100 cm intervals, as seen in Fig. 9.

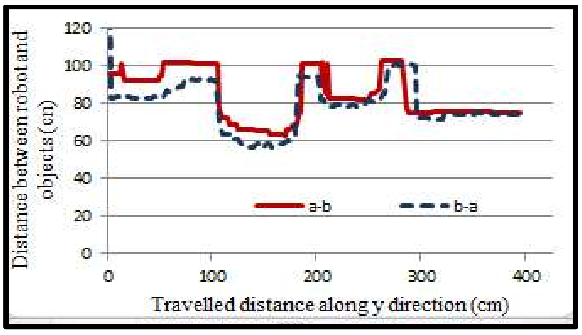

Figure 9. Hysteresis errors obtained from the ultrasonic sensors due to the change in the direction of the robot's movement.

$$Max.H\ (\%fsd) = \frac{max(O\ I\downarrow - O I\uparrow)}{Output\ span} \qquad (11)$$

O I ↓: The measured distance value for when the robot moves from point B to A.

O I ↑: The measured distance value for when the robot moves from point A to B.

The output span is 100 cm.

6. CONCLUSIONS

In this study, a low-cost FPGA-based non-autonomous differentially driven wheeled mobile robot was designed so as to build 2D maps for indoor environments without any pre-exploration.

The sensors used in indoor mapping systems tend to be similar for all robots; the factor that determines the cost of a robot is the control unit, except for when LIDAR is used. Unlike previous studies [32, 41], our system can be considered to be low-cost. In the indoor mapping application by Mirowski et al. [33], the robot is also classified as low-cost ; however, the robot designed in that study relies on using a Kinect depth camera from Microsoft Inc., which is limited by a narrow field of view and has a short range (about 50 cm to 5 m). Compared to this, even if the costs are close to each other, the system proposed in this paper has a wider range. In addition, filtering was not used in our study due to the features of the AHRS sensor (i.e., a built-in sensor that can automatically compensate for the interference of temperature drift, noise, and external magnetic fields).

After the hardware and software components of the MappingBot were successfully set up, the accuracy of the system, such as its linearity, hysteresis, and repeatability, were all tested. If any of the factors that contribute to the measurement errors are defined correctly, and if these errors seem repeatable, then they can be easily removed. By applying an error correction algorithm on the raw data, we found that the average overall error along the y-axis is 0.01%, whereas the error along the x-axis was found to be 2.2%. The maximum error resulting from the encoder was calculated as being 0.021% and the maximum error from the ultrasonic sensors was 5.20%.

The majority of studies [1, 2, 7, 9] on mapping have not conducted any such a separate accuracy assessment for both x and y dimensions. A similar real time solution for indoor mobile mapping was recently published by Niu Xiaoji et al [23]. In their multi-sensor (LIDAR/IMU/Camera) system, two types of Extended Kalman Filter (post-processed and online) were used and their accuracy was compared. Rms error of their selected feature points had been reported as 5.6 and 7.32 cm, respectively. Due to the evaluation given in terms of cm instead of percentage; an accurate comparison cannot be done between that paper and ours. In another real time work presented by W. Hess which uses a low cost laser distance sensor computes the map with a quantitative error of max 0.8% with a resolution of 5 cm [24]. On the other hand, their paper emphasizes on the calculation and building a map by using their own algorithm from collected data rather than designing a new mapping hardware. In the study presented by Gong et al. [29], it was emphasized that the robot locates and maps itself with a positioning accuracy of 10–30 mm. However, this is not an accurate indicator, since the dimensions of the reference area upon which the map was built is not given. For this reason, a percentage accuracy comparison cannot be made between the two. Therefore, this study could be the first that makes such an assessment in order to provide information about the overall accuracy of a mapping system.

For further research, we will using new algorithms such as, SLAM or Monte Carlo positioning approaches in order to minimize the errors arising from the wheel odometry method. Additionally we will use different sensors, such as LIDAR, laser scanners, or cameras, in order to minimize the errors arising from the ultrasonic sensors.Thus, image processing technologies could be used in distance measurements by adding a camera to the mapping system. In addition, methods that could decrease the hysteresis behavior of the mapping system to below 10% will be prioritized. Future studies will aim to transform a non-autonomous robot into an autonomous one.

7. Acknowledgments

This study has been supported by Scientific Research Project of Marmara University (Project No. FEN-C-YLP-060911-0281).

8. REFERENCES

[1] Link, J. A. B., Smith, P., Viol, N., & Wehrle, K. (2011, September). Footpath: Accurate map-based indoor navigation using smartphones. In Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on (pp. 1-8). IEEE.

[2] Hil Senbeck, S., Bobkov, D., Schroth, G., Huitl, R., & Steinbach, E. (2014, September). Graph-based data fusion of pedometer and WiFi measurements for mobile indoor positioning. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (pp. 147-158). ACM.

[3] Negenborn, R. (2003). Robot localization and Kalman filters (Doctoral dissertation, Utrecht University).

[4] Lee, B. G., & Chung, W. Y. (2011). Multitarget three-dimensional indoor navigation on a PDA in a wireless sensor network. IEEE Sensors Journal, 11(3), 799-807.

[5] Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. AAAI/IAAI, 1999(343-349), 2-2.

[6] Zhao, H., & Wang, Z. (2012). Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended kalman filter for data fusion. IEEE Sensors Journal, 12(5), 943-953.

[7] Fu, G., Zhang, J., Chen, W., Peng, F., Yang, P., & Chen, C. (2013). Precise localization of mobile robots via odometry and wireless sensor network. International Journal of Advanced Robotic Systems, 10(4), 203.

[8] Biswas, J., & Veloso, M. (2010, May). Wifi localization and navigation for autonomous indoor mobile robots. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 4379-4384). IEEE.

[9] Ni, L. M., Liu, Y., Lau, Y. C., & Patil, A. P. (2004). LANDMARC: indoor location sensing using active RFID. Wireless networks, 10(6), 701-710.

[10] Kiers, M., Bischof, W., Krajnc, E., & Dornhofer, M. (2011, September). Evaluation and improvements of an rfid based indoor navigation system for visually impaired and blind people. In 2011 International Conference on Indoor Positioning and Indoor Navigation; Paper, Guimarães, Portugal (Vol. 16).

[11] Yenilmez, L., & Temelta, H.(2001) Mobil Robotlarda Ultrasonik Duyaçlarla Gerçek Zamanda Konum Kestirimi.Istanbul University Engineering Faculty Journal of Electrical & Electronics, 1(2), 249-255. (in Turkish)

[12] Fujimoto, M., Nakamori, E., Inada, A., Oda, Y., Wada, T., Mutsuura, K., & Okada, H. (2011, September). A broad-typed multi-sensing-range method for indoor position estimation of passive RFID tags. In Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'2011).

[13] Han, S., Lim, H., & Lee, J. (2007). An efficient localization scheme for a differential-driving mobile robot based on RFID system. IEEE Transactions on Industrial Electronics, 54(6), 3362-3369.

[14] Malu, K., & Jharna Majumdar, S. (2014). Kinematics, Localization and Control of Differential Drive Mobile Robot. Global Journal of Research In Engineering, 14(1).

[15] Altun, K., & Barshan, B. (2012). Pedestrian dead reckoning employing simultaneous activity recognition cues. Measurement Science and Technology, 23(2), 025103.

[16] Hamerlain, F., Floquet, T., & Perruquetti, W. (2014). Experimental tests of a sliding mode controller for trajectory tracking of a car-like mobile robot. Robotica, 32(01), 63-76.

[17] Ojeda, L., & Borenstein, J. (2004). Methods for the reduction of odometry errors in over-constrained mobile robots. Autonomous Robots, 16(3), 273-286.

[18] Korkmaz, H., Cosgun, E. Odometry error recovery by using a PID controller in the case of linear displacement of 4-wheeled indoor mapping robot (in Turkish) TOK2013-National Automatic Control Meeting and Exhibition. Malatya İnönü Üniversitesi, Türkiye, 2013 Sep.

[19] Maimone, M., Cheng, Y., & Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. Journal of Field Robotics, 24(3), 169-186.

[20] Kuramachi, R., Ohsato, A., Sasaki, Y., & Mizoguchi, H. (2015, December). G-ICP SLAM: An odometry-free 3D mapping system with robust 6DoF pose estimation. In Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on (pp. 176-181). IEEE.

[21] Lang, H., Wang, Y., & de Silva, C. W. (2008, September). Mobile robot localization and object pose estimation using optical encoder, vision and laser sensors. In Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on (pp. 617-622). IEEE.

[22] Kottmeier, S., Müller, S., Schmidt, T., Zajac, K., & Schmalbach, M. (2011). A High-Precision, High-Reliability Binary Rotary Encoder Using Hall-Effect Sensors. Proc. Of the 14th ESMATS.

[23] Niu, X., Yu, T., Tang, J., & Chang, L. (2017). An Online Solution of LiDAR Scan Matching Aided Inertial Navigation System for Indoor Mobile Mapping. Hindawi Publishing Corporation Mobile Information Systems.

[24] W. Hess, D. Kohler, H. Rapp, and D. Andor, Real-Time Loop Closure in 2D LIDAR SLAM, in Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016. pp. 1271–1278.

[25] Esenkanova, J., İlhan, H. O., & Yavuz, S. (2013). Pre-Mapping system with single laser sensor based on gmapping algorithm. IJOEE: Int. Journal of Electrical Energy, 1(2), 97-101.

[26] Eliazar, A., & Parr, R. (2003, August). DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In IJCAI (Vol. 3, pp. 1135-1142).

[27] Ponce-Cruz, P., Molina, A., & MacCleery, B. (2016). Fuzzy Logic Type 1 and Type 2 Based on LabVIEWTM FPGA (Vol. 334). Springer.

[28] Orlowska-Kowalska, T., & Kaminski, M. (2011). FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system. IEEE transactions on Industrial Informatics, 7(3), 436-445.

[29] Gong, Z., Li, J., & Li, W. (2016, July). A low cost indoor mapping robot based on TinySLAM algorithm. In Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International (pp. 4549-4552). IEEE.

[30] Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Mobile robot positioning-sensors and techniques. Naval Command Control And Ocean Surveillance Center Rdt And E Div San Diego Ca.

[31] Fulmek, P. L., Wandling, F., Zdiarsky, W., Brasseur, G., & Cermak, S. P. (2002). Capacitive sensor for relative angle measurement. IEEE Transactions on Instrumentation and Measurement, 51(6), 1145-1149.

[32] Hide, C., Moore, T., & Smith, M. (2004, April). Adaptive Kalman filtering algorithms for integrating GPS and low cost INS. In Position Location and Navigation Symposium, 2004. PLANS 2004 (pp. 227-233). IEEE.

[33] Mirowski, P., Palaniappan, R., & Ho, T. K. (2012, April). Depth camera SLAM on a low-cost WiFi mapping robot. In Technologies for Practical Robot Applications (TePRA), 2012 IEEE International Conference on (pp. 1-6). IEEE.

[34] Sileshi, B. G., Oliver, J., Toledo, R., Gonçalves, J., & Costa, P. (2016). On the behaviour of low cost laser scanners in HW/SW particle filter SLAM applications. Robotics and Autonomous Systems, 80, 11-23.

[35] Kohlbrecher, S., Meyer, J., Graber, T., Petersen, K., Klingauf, U., & von Stryk, O. (2013, June). Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots. In RoboCup (pp. 624-631).

[36] Szöcs, D., Pană, T., Feneşan, A., & Vese, I. (2012, October). Error and drift attenuation for wheel slip measurement of a prototype electric vehicle. In Electrical and Power Engineering (EPE), 2012 International Conference and Exposition on (pp. 64-69). IEEE.

[37] Monmasson, E., Idkhajine, L., Cirstea, M. N., Bahri, I., Tisan, A., & Naouar, M. W. (2011). FPGAs in industrial control applications. IEEE Transactions on Industrial informatics, 7(2), 224-243.

[38] Korkmaz, H., Cosgun, E., Bal, S. A., & Toker, K. (2014, April). Developing a graduate level embedded system programming course content by using blended programming methodologies: Text-based and graphical. In Global Engineering Education Conference (EDUCON), 2014 IEEE (pp. 1010-1015). IEEE.

[39] Falcon, J. S., & Trimborn, M. (2006, June). Graphical programming for field programmable gate arrays: applications in control and mechatronics. In American Control Conference, 2006 (pp. 7-pp). IEEE.

[40] Kamenar, E., & Zelenika, S. (2013, May). Micropositioning mechatronics system based on FPGA architecture. In Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on (pp. 125-130). IEEE.

[41] MacCleery, B., & Kassas, Z. M. (2008). New mechatronics development techniques for FPGA-based control and simulation of electromechanical systems. IFAC Proceedings Volumes, 41(2), 4434-4439.

[42] NI White Papers, Graphical System Design Basics: Accelerating Development Time and Bringing Embedded Design to the Masses, Publish Date: Oct 19, 2012. http://www.ni.com/white-paper/3392/en/

[43] Pololu Robotics and Electronics https://www.pololu.com/product/1217 (22.06.2017)

[44] NI White Papers, Quadrature Encoder Example DAQ Personality, Publish Date: Aug 24, 2016. (22.06.2017)

[45] Product manual, https://www.dfrobot.com/wiki/index.php/URM37_V4.0_Ultrasonic_Sensor_(SKU:SEN0001) (22.06.2017)

[46] High-precision positioning inertial navigation system Product page, http://feiyutech.en.alibaba.com/product/528618009801939455/FY_2000B_AHRS.html (22.06.2017)

[47] Digi XBee® ZigBee Product page, " Embedded ZigBee and Thread-ready RF modules provide OEMs with a simple way to integrate mesh technology into their application", https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee (22.06.2017