Design and Implementation of a Two-Dimensional Ultrasonic Radar using FPGA

Hamed Seyedroudbari, Xiaoao Feng, Grant Crongeyer, Saba Janamian,
Andrew Zaragoza, Sohrab Azarakhsh, Rifaat Kouaider, *Member, IEEE*
Advisor: Shahnam Mirzaei, Ph.D.
California State University, Northridge
November 16, 2018

***Abstract*—In this paper, an Ultrasonic Radar is introduced that is capable of detecting an object in 3D space and mapping the detected object's position on a 2D graph. Using time-of-flight methodology, an ultrasonic sensor proves to be a cheap and easy way of detecting an object. In fact, the most crucial problem for mobile and robot navigation systems is their ability to detect and localize obstacles. The more advanced systems in this realm use laser range finders and vision to solve this issue. However, laser scanners are expensive, only scan a single plane, and the amount of data they produce is immense, which makes data processing more time-consuming and costly. Ultrasonic sensors provide a cost-effective way of implementing a complex detection system. In this project, by incorporating two ultrasonic sensors, it is possible to scan a wider space and pinpoint the precise location of the detected object on a 2D interface. Due to their inherent nature, the ultrasonic sensors may be prone to a lot of noise interference, especially in crowded areas, in which case the transmitted beam may reflect off of several objects and lead to the reception of an unstable reading at the receiver. Therefore, this project is intended for and tested in small spaces with only one or a couple objects or people around.**

***Index Terms*—Client-server systems, FPGA, Microcontrollers, Ultrasonic imaging**

I. Introduction

THIS ultrasonic radar project uses two ultrasonic radars to detect the distance away from a single object. The distance readings received from the two sensors are then used to perform calculations on a Xilinx FPGA/SoC programmable board. Assuming the formation of a triangle with two sensor beams pointing toward the detected object, as well as a solid axis holding them together, geometric calculations are done to calculate the distance of the detected object from the center of the rotating axis. The calculated distance is then passed to the processing system of the FPGA board, on which Linux is running, sent to a web server, and displayed on a 2D graphical interface. The system will detect an object in 3D space and display it on a 2D graphical interface. This idea is shown in Fig. 1.

A. Background

The idea for this project rose from mobile and robot navigation systems and their ability to detect obstacles. These days, the more advanced object detection and radar systems use expensive laser range finders and computer vision to solve these problems. Laser range finders are only able to scan a single plane and have a very narrow beam to detect objects. In addition, computer vision systems produce a large amount of data that requires costly and time consuming processing. By using ultrasonic sensors, while restricted to a detection range of 15 ft, the ultrasonic radar will provide a smaller and more cost-effective version of these detection systems.

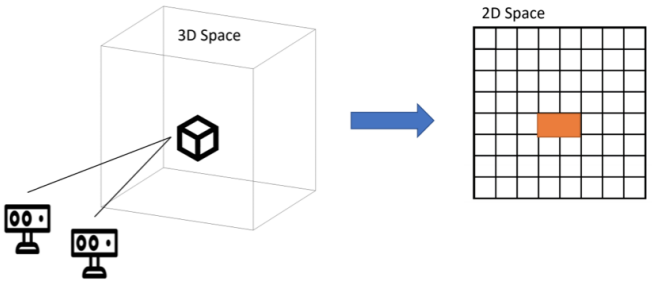

Fig. 1. Diagram displaying the desired final functionality of the ultrasonic radar.

B. Requirements

The ultrasonic radar was to detect objects within a 15-foot proximity range. The radar system would be able to rotate 180 degrees using a servo motor to have detection coverage in every direction for a wider viewing range. Detecting an object in 3D space, two ultrasonic sensors were to be used to map the position of the detected object on a 2D graph. Ultrasonic sensors are prone to a lot of noise interference, especially in crowded areas, where the sensor may receive a beam that has been reflected off several different objects. This problem is shown in Fig. 2.

Considering the system to be tested in open areas with not many interfering objects around or in classrooms, an ideal distance between the two sensors was decided to be 3 ft. The 3-foot distance was accounted for in the length of the axis that was rotating the radar system. The length and weight of the axis must have been chosen while considering the power and torque of the servo motor on which it will be positioned. Regarding graphing and displaying of data, the goal was to work with a development board that would allow the installation an operating system and have it directly connect to a network. In other words, the idea was to eliminate the need for a secondary machine acquiring data and communicating

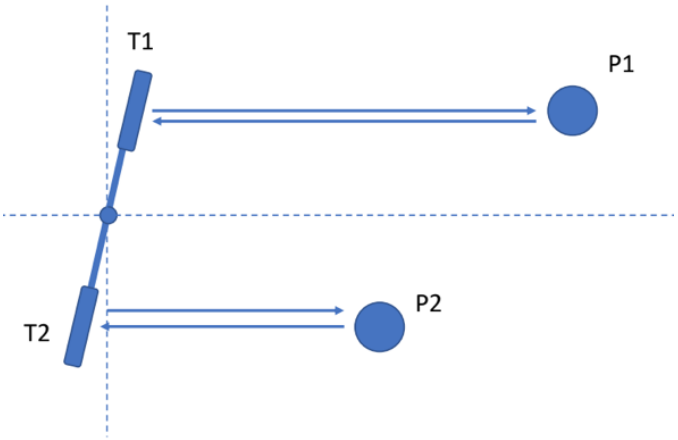

Fig. 2. Sensors receiving beams that could be reflecting off several objects.

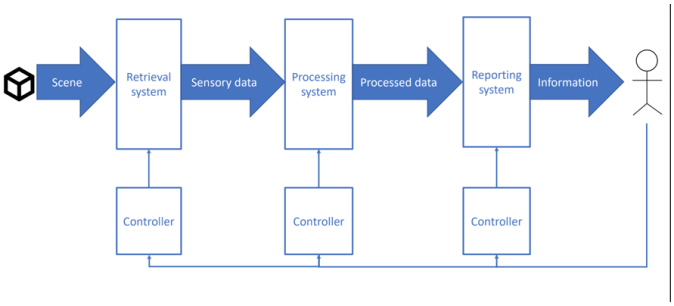

Fig. 3. General data flow diagram of the ultrasonic radar.

with the web server. To view a general the data flow diagram of the project, please refer to Fig. 3.

II. Design Approach and Calculation

A. Ultrasonic Sensor

1) Principle: Ultrasound is a mechanical wave which has a frequency range higher than the range a human ear can sense. Some ultrasound may even exceed 10 MHz, depending on the application it is used for. Ultrasonic sensors use time of flight (TOF) of a transmitted pulse to measure the distance to an object. In simpler terms, the sensor transmits a pulse at a certain frequency and waits for to echo back to the sensor. The TOF can be evaluated using $x = c \times T_0/2$, in which c is the speed of sound, and T_0 is the time interval between pulse transmission and echo reception. Ideally, it is assumed that the echo signal is undistorted and attenuated by a constant coefficient. However, in the real world, multiple factors such as humidity, temperature, and applied voltage can distort the signal.

In this project, the LV-MaxSonar-EZ1 was used as the ultrasonic sensor. This sensor, shown in Fig. 4, operates at a frequency of 42 kHz with a 5 V DC input voltage. The subsection below will explain how the sensor actually operates. It is important to note that the LV-MaxSonar-EZ1 transmits a beam which is 33 degrees wide. This is important when two sensors are trying to detect the same object. With a transmitted beam as wide as 33 degrees, each sensor, although intended

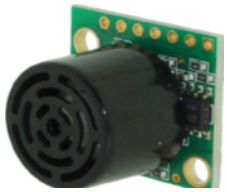

Fig. 4. Image of LV-Maxsonar-EZ1 used to measure distance to surrounding objects.

to detect the same object, may potentially detect two different objects and hence, read different distance values.

2) Operation: Initially, the LV-MaxSonar-EZ1 requires 250 ms to power-up. When a high voltage is asserted at the sensor's RX pin, the sensor will start a 49 ms calibration period and allow range readings to be taken every 49 ms after that. The first reading, however, will take an additional 100 ms to be processed. When the RX pin is at a logic high value, the sensor will transmit its pulse, making the pulse width pin (PW) go to a logic high value. The PW pin will be set to low logic value, upon detecting a target ,or remain at a high logic value for up to 37.5 ms if no target is detected. Fig. 5 below shows a diagram of the process the LV-MaxSonar-EZ1 must go through to obtain a single sensor reading.

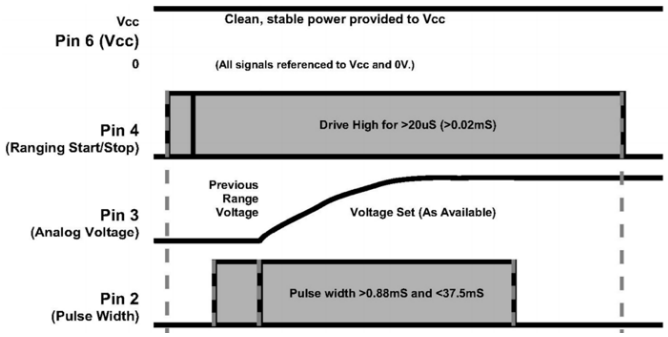

Fig. 5. Diagram showing how LV-MaxSonar-EZ1 operates to obtain a single distance reading.

B. Zedboard

The two LV-MaxSonar-EZ1 sensors are wired to a re-programmable FPGA board as peripheral modules. This makes communication between the sensors and the board more convenient and allows for easy access to sensor readings and manipulation through geometric calculations. With a stream of sensor readings being constantly generated, there was a need for a machine that would be able to consistently process the streaming data and perform mathematical calculations at a high rate. It was also necessary to communicate the calculated data to the server to map an object's distance and position in real time.

For these reasons, a Zedboard Zynq-7000 ARM/FPGA SoC development board was used. The Zedboard, as shown in Fig. 6, was an ideal choice since it incorporated both an FPGA and a Zynq processor. To be able to consistently process the incoming stream of data, the FPGA helped to fulfill the parallelism requirement as well as the pipelining of the

Fig. 6. Image of a Zedboard Zynq-7000 ARM/FPGA SoC development board.

different stages. The highly clocked processor allowed the installation of the Linux operating system on the board and connect the Zedboard to a network via ethernet. This was helpful in establishing a direct connection from the Zedboard to the web server without going through another machine.

As mentioned earlier, the search for a board with several peripheral modules and communication interfaces was desirable for the project's purposes. As shown below in Fig. 7, the Zedboard has a plethora of peripheral modules and interfaces. On the PL side, which consist of the FPGA, the following peripheral features were used: Primary JTAG, GPIOs (LEDs, Switches), PMODs. The JTAG was primarily used to program the FPGA to acquire distance values from sensors and perform calculations on them. The LEDs were configured for debugging purposes and for validation of distance measurements and calculations. The switches were used to control the servo motors which rotated the LV-MaxSonar-EZ1 sensors.

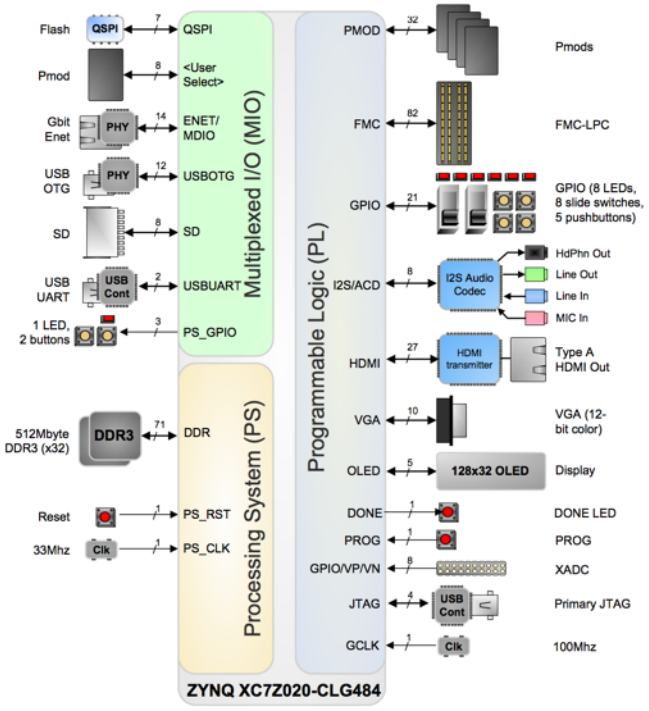

Fig. 7. Image displaying peripherals and interfaces on a Zedboard.

C. Server

To serve radar's web application a Python Tornado web server was used. Tornado is a non-blocking web framework that can manage a large number of connections to the server. The Tornado web framework has the capability of supporting WebSocket and long polling applications that require streaming and long established connections. The Tornado web server will accept http request from the application and will broadcast it to all the applications connected to the web server via WebSocket. The webserver setup block diagram is shown in Fig. 8.

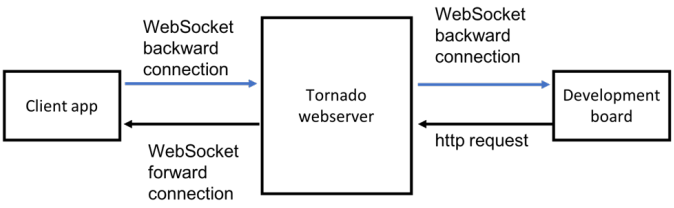

Fig. 8. Webserver setup schematic.

The graphical part of the application is written in P5.JS which is a graphical library for creating HTML5 graphical applications. The P5.JS is written in JavaScript language and can support streaming and rendering the data in the client application.

D. Physical setup

1) Bar: The bar refers to the 3-foot beam that is running across the top of the Zedboard. The purpose of this bar is to hold the two TowerPro SG90 servos while being able to be rotated along the vertical axis with the aid of the Lewansoul LD-3015M servo motor. The bar was previously made out of wood and was unable to hide all of the wires and cables. This semester, the new version of the bar will be made out of polyvinyl chloride (PVC) as well as some aluminum brackets. This upgrade will create a cleaner cable management while accomplishing the aforementioned goal of holding the two servos on each end while being able to rotate.

2) Casing: This semester, the casing is upgraded on the TowerPro SG90 servos as well as on the Zedboard. The previous iteration did not have a case for the servo motors positioned on the bar. They were held onto the bar with metal clamps, which did not have the best cable management nor aesthetic design. This semester, SolidWorks was used in order to create a casing that would be able to house the servo motor as well as be placed onto the bar easily, all with cable management in mind. To measure the dimensions of the servos, a vernier caliper is used to insure accuracy and precision. With this 3D printed case, it is possible to have the same flexibility as before while keeping cables out of sight and have the motors be more secure. Fig. 9 is the Solidworks servo case design.

As for the casing on the Zedboard itself, there were some minor complications. The casing needed to be transparent so others would be able to see the Zedboard setup while having holes for necessary wires and connections. In order to achieve

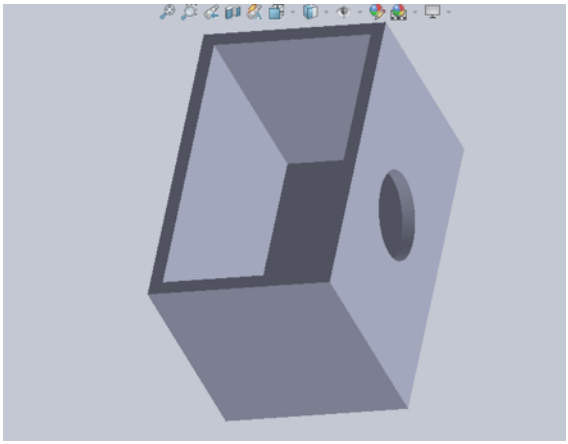

Fig. 9. Solidworks Servo Case Design.

the requirements that were set, only two plausible options were a case made of either plastic or glass. The proper tools were not available to cut glass, so it was decided to use a plastic casing. The dimensions were measured with a ruler and allowed some extra space of working room, particularly in the height. The height of the Zedboard case is about 10" in order to account for the Lewansoul LD-3015M servo motor that will be placed on the top of the case.

3) Positioning of sensors: The two ultrasonic sensors will be placed on either side of the bar, 3' away from each other. This value of 3' will always be a known constant value, therefore aiding in the detection and ranging calculations.

E. Servo motors

To allow for a wider viewing angle and detect objects over a larger area, the LV-MaxSonar-EZ1 sensors were placed on TowerPro SG90 servo motors. The goal of using servos was to be able to control the angle at which the sensors are facing in real time. This was implemented through the server as well as using the DIP Switches located on the Zedboard. The TowerPro servo motor is controlled by predefined pulse width that will trigger the mechanical elements of the servo to rotate. This servo rotates from -90 degrees to 90 degrees with a proportional dependency on the pulse width that triggers it. For this servo, a pulse width of 1-2 over a pulse width period of 20 ms is enough to rotate the servo anywhere within the -90 degree to 90 degree range. Fig. 10 shows exactly this.

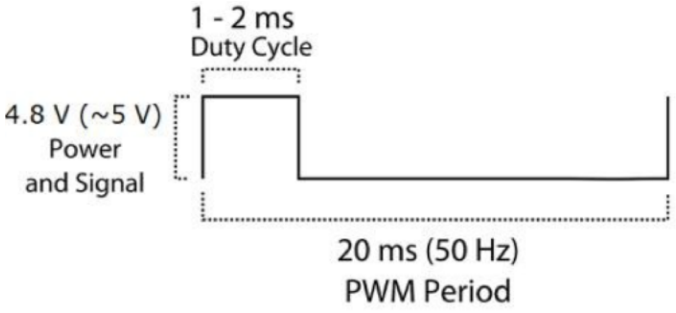

Fig. 10. Pulse width and period that is required to rotate the servo.

In addition to the two servo motors on the two ultrasonic sensors, a third servo, the Lewan Soul LD-3015MG was in fact used to rotate the middle axis holding the sensors. This servo motor was added to maximize the consistent coverage of the radar in all directions. Similar to the TowerPro SG90, the LD-3015MG has a 20 ms pulse period, however, it is able rotate a total of 270 degrees. The wider rotating range is due to the servo motor's ability to accept a wider range of pulse widths. This servo motor in particular accepts pulses from 0.5 ms to 2.5 ms.

F. Design approach

The design of the ultrasonic radar started with the individual components. The goal of the project was to be able to map an object in 3D space on a 2D graph. In order to do that, two sensors are required to ensure that both sensors are detecting the same object. Due to the instability and bouncy nature of an ultrasonic pulse, it was decided to test the device in an open area, preferably confined, with only a few objects present.

To make the physical setup suitable for a confined area with few present objects. The two sensors were placed on a three foot stick, three feet apart. The two sensors made calculating the distance and mapping it on a 2D graph much simpler. In addition, to maximize the detection coverage of the radar, the individual sensors as well as the bar were each placed on top of servo motors that rotate at least 180 degrees.

To facilitate the communication of data from the Zedboard to the online server, Linux was installed as an operating system on the processor on the Zedboard. By doing this, the ethernet interface was ready to be configured and a simpler communication protocol between the FPGA and processor was implemented to send calculated values from the FPGA the processor, and from the processor to the server.

G. Calculations

Using two Ultrasonic sensors and placing them on an axis, three feet apart, and assuming that the two sensors are detecting the same object, it's possible to form a triangle. Since the sensors' transmitted beam is 33 degrees wide, the triangle formed may not necessarily be an isosceles triangle. For this reason, a more complex set of geometric calculations had to be used to account for situations when the sensors detect objects at different parts of their viewing angles. Knowing the values of a, b and c from the sensor readings and the constant length of the axis, the area of the formulated triangle, as shown in Fig. 11, can be calculated using Eq. (1) and Eq. (2) .

$$s = \frac{a+b+c}{2} \tag{1}$$

$$Area = \sqrt{s(s-a)(s-b)(s-c)} \tag{2}$$

After calculating the area of the triangle, Eq. (3) and Eq. (4) can be used to calculate the height of the triangle as well as the median, shown as r in Fig. 11, respectively. Finding these values will allow to calculate the reference angle of the median with respect to the rotating axis.

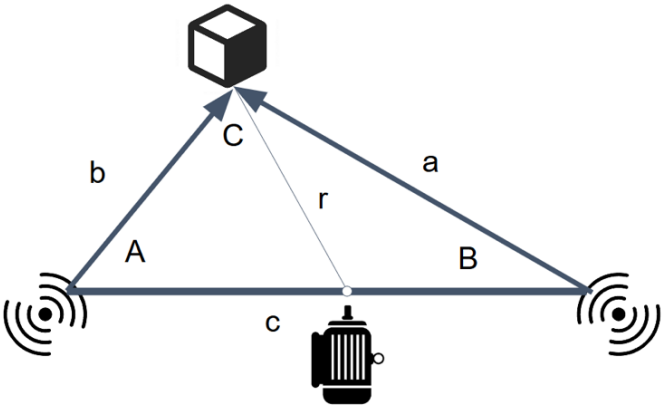

Fig. 11. Diagram displaying a possible situation when detecting an object.

$$h = \frac{2 \times Area}{c} \tag{3}$$

$$m = \frac{\sqrt{2a^2 + 2b^2 - c^2}}{2} \tag{4}$$

$$\theta = \sin^{-1}(\frac{h}{r}) \tag{5}$$

Due to the wide beam width the sensors are transmitting, angles A and B, as shown in Fig. 11, may not be the actual angles the servo motors are positioned at. To account for a possibility in angle changes, Eq. (6) and Eq. (7) are used to calculate the respective angle of each sensor with respect to the rotating axis, using only the side lengths of the formulated triangle.

$$\cos A = \frac{-a^2 + b^2 + c^2}{2bc} \tag{6}$$

$$\cos A = \frac{-b^2 + a^2 + c^2}{2ac} \tag{7}$$

These distance values, median, its angle, as well as the respective angle of each sensor with respect to the axis makes it possible to plot the distance and position of the detected object on a polar and coordinate graph. Images of the polar and coordinate graphs are shown Fig. 12, and Fig. 13, respectively.

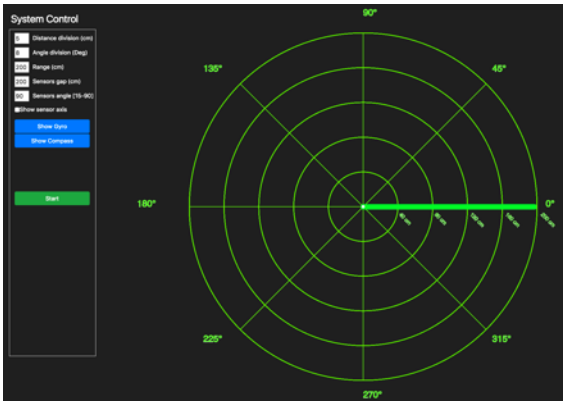

Fig. 12. Diagram displaying polar graph while detecting an object.

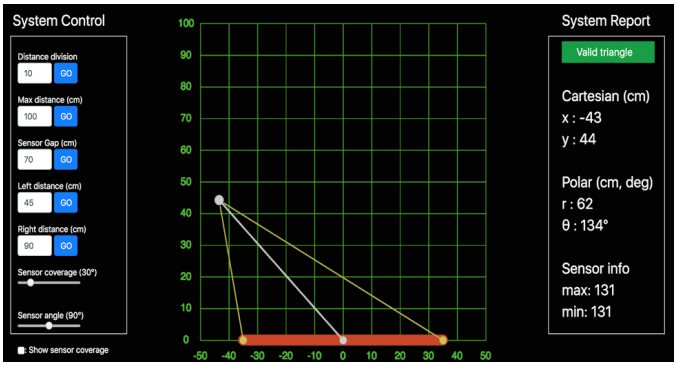

Fig. 13. Diagram displaying coordinate graph while detecting an object.

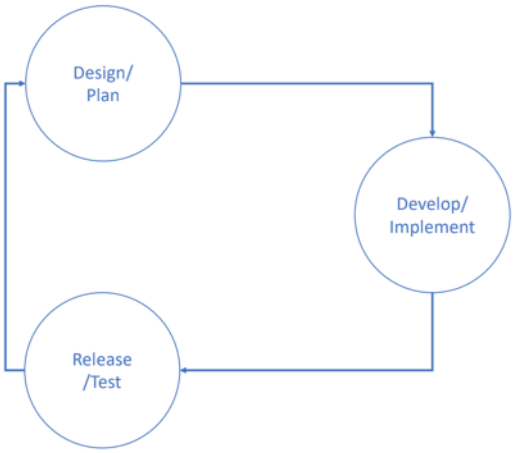

Fig. 14. Diagram explaining the process involved in planning, designing, and testing each individual module.

H. Testing

The testing which was incorporated during the project involved single unit testing as well as testing multiple units combined. Using VHDL, the servo motors were configured using VHDL in addition to the geometric calculations which were used to calculate the distance to the detected object. Each operation (e.g. square rooter, divider, servo controller) was initially planned and designed as its own individual module. Each individual module was tested to verify correct functionality before being integrated with other modules. The testing process involved with each developed module is shown in the Fig. 14.

The same methodology was followed for the installation of Linux on the processor of the Zedboard. Linux was first installed on the Zedboard. After installation, the communication between the FPGA and the processor as well as from the processor to the server were tested separately. Once, the functionality of Linux was verified, it was integrated with the other parts of the project.

Once the individual modules were designed and tested, the entire system was tested. Objects were placed at various distances from the sensors and the sensors were placed at different angles to test the resulting final calculated distance appearing on the 2D graphical interface.

III. Future plans

With the rapid advancement in the autonomous vehicles the need for object location detection has been dramatically increased. In future work it would be preferable to find the location of an object in the 3D space. The plan for the future work of the project is to compute the depth of an object in an image by using the stereo vision. For the stereo vision we will use two cameras. The cameras will take a picture from the same object from slightly different angle. If only one camera was used it would not be possible to detect the distance of the object from the camera since there is only the x and y coordinate of the object and not the depth or z coordinate. Fig. 15 shows the one camera configuration where x is the image plane and O is the optical center of the camera. As seen in the figure, both objects Q and P project into the same location. With one camera setup this will happen for each point along the same line of projection.

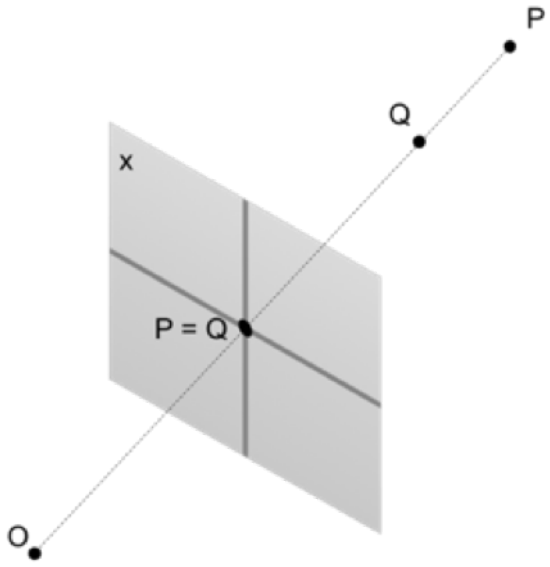

Fig. 15. One camera setup illustration. X is the image plane and O is the optical point of the camera.

With two cameras it is possible to calculate the depth by using triangulation. The requirement for the triangulation calculation is to find homologous (corresponding) points in the two pictures. Fig. 16 shows the two camera setup. From the figure objects P and Q have different corresponding points on the right image. One of the computationally challenging tasks of triangulation calculation is finding the corresponding pixels on the two images. The 2D search along the x and y direction of the two images will cause large computational overhead which can be impractical in real time processing.

One of the techniques to eliminate searching in the entire image for finding corresponding points is the epipolar constraint. Consider the schematic shown in Fig. 17. The plane PLR is called the epipolar plane. The epipolar constraint states that corresponding points on line PL passing through plane x1 lay on line l' on the x2 plane. To find any point on the PL line we only need to search on line l'. The l' line is called the epiline.

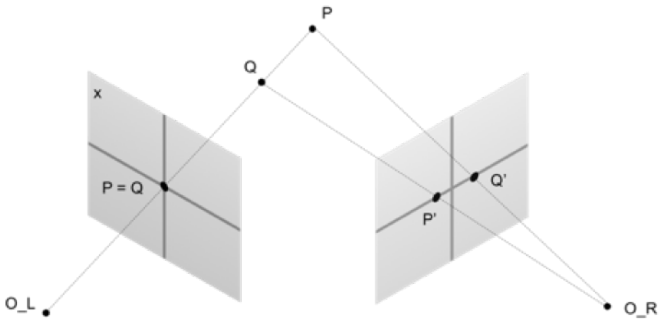

Fig. 16. Two cameras setup for stereo vision.

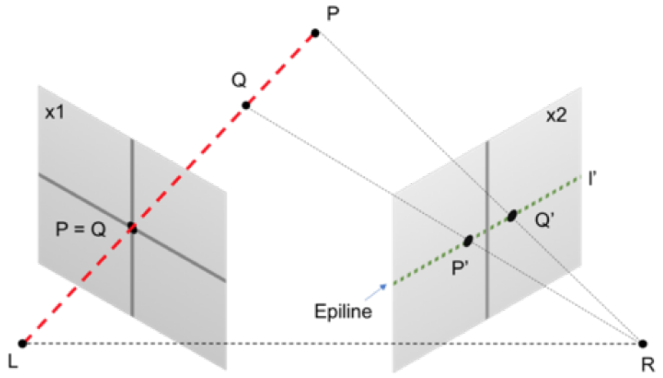

Fig. 17. The line l' is called the epiline and the plane PLR is called the epipolar plane.

The disparity map is the difference between to corresponding points on the epiline. As objects get closer to the cameras, they will have larger disparity and as they get farther from the cameras they will have smaller disparity values. The image created from the disparity of two image is called the disparity map and can be shown as colored map or gray scaled. Fig. 18 and 19 show the concept of disparity. In Fig. 18, object a has been captured by two cameras. The corresponding location of object a on a hypostatical pictures XL and XR are shown. The corresponding points on the object have two units of difference.

Fig. 19 shows the setup for capturing two objects aligned at the center of the setup. The object b is farther than object a to the cameras. The b object will have smaller disparity since it is farther from the cameras.

To find the distance D of the object a from the cameras in Fig. 18 the similar triangles rule can still be applied. In the calculation, we assumed that the two cameras are aligned and the focal point of the two cameras is calibrated to the value f. The image target a will be at distance x_L in the left camera and at the distance x_R in the right camera. The distances from the center of the setup to the focal line of the two cameras are b_L and b_R. Since it is assumed the object is at the center of the setup, the two distances b_L and b_R will be identical and their summation will be equal to value b. By using triangulation, we get the Eq. (8) and Eq. (9). Since $b = b_L + b_R$, then we get Eq. (10) and Eq. (11).

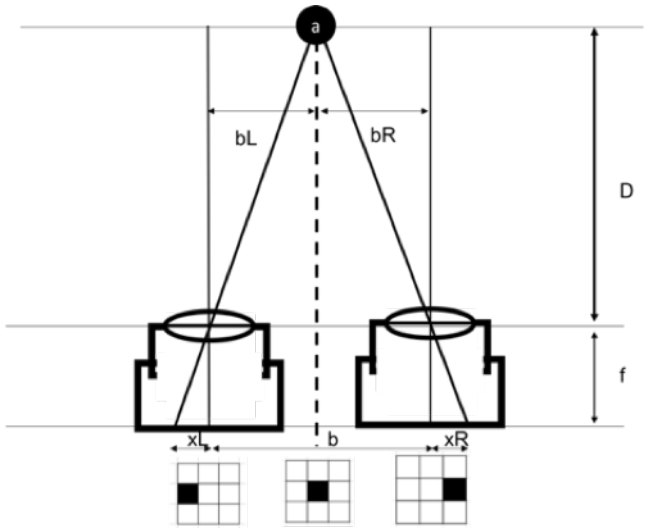

Fig. 18. Disparity map of one object captured by two cameras.

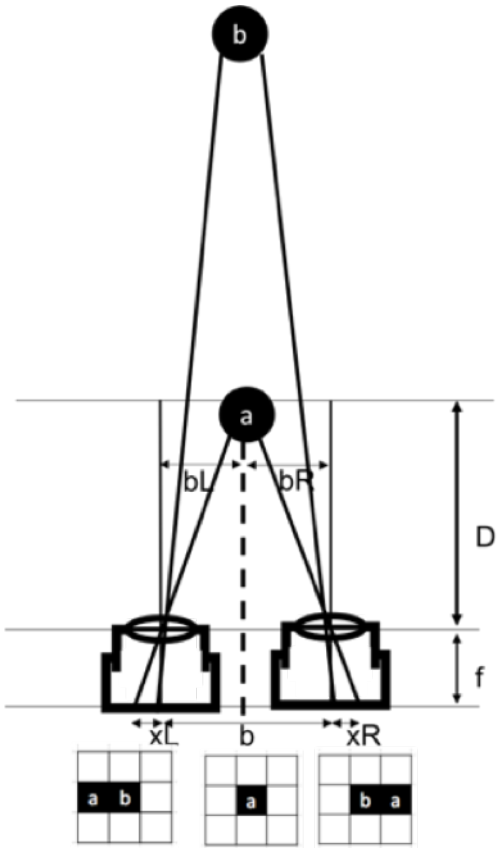

Fig. 19. Disparity map of two object captured by two cameras.

$$\frac{b_L}{D} = -\frac{x_L}{f} \tag{8}$$

$$\frac{b_R}{D} = \frac{x_R}{f} \tag{9}$$

$$b = \frac{D}{f}(x_R - x_L) \tag{10}$$

$$D = \frac{b \times f}{x_R - x_L} \tag{11}$$

It is currently working on the implementation of the OV7670 camera module on the ZedBoard FPGA development board. The OV7670 is a CMOS image sensor with low power consumption, as shown in Fig. 20. The camera is capable of providing the full functionality of a single chip VGA camera. The camera module consists of 18 pins.

Fig. 20. OV7670 camera module.

Fig. 21 shows the schematic diagram of OV7670 camera based system. The camera module can work with 3.3V power supply. The camera also requires an external oscillator to provide clocking to the module. Pin XCLK can be used to supply the clock. The module is controlled with I^2C signal. By configuring the proper register and supplying I^2C clock to pin PCLK the camera can provide data to pins d0 to d9. The data can be synchronized with the host with signals HREF or VSYNC.

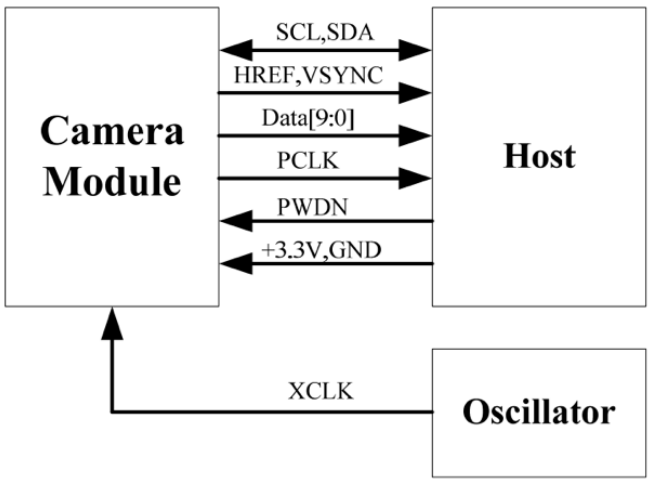

Fig. 21. Schematic diagram of OV7670 camera based system.

In order to show the result of the 3D mapping a web application was created that is capable of showing objects in the 3D space. The application uses Three.js which is a wrapper library for the WebGL framework. The application is a JavaScript based application that can receive a stream of data via a WebSocket connection. The application can receive the x, y, and z coordinate of the object and can map them on the 3D plane shown in the Fig. 22. The application can also receive control inputs for the size of the image in the x, y and z direction, the rotation orientation of the object in the 3 coordinates and the camera location looking at the object. The application also can show and update multiple objects on the 3D plane.

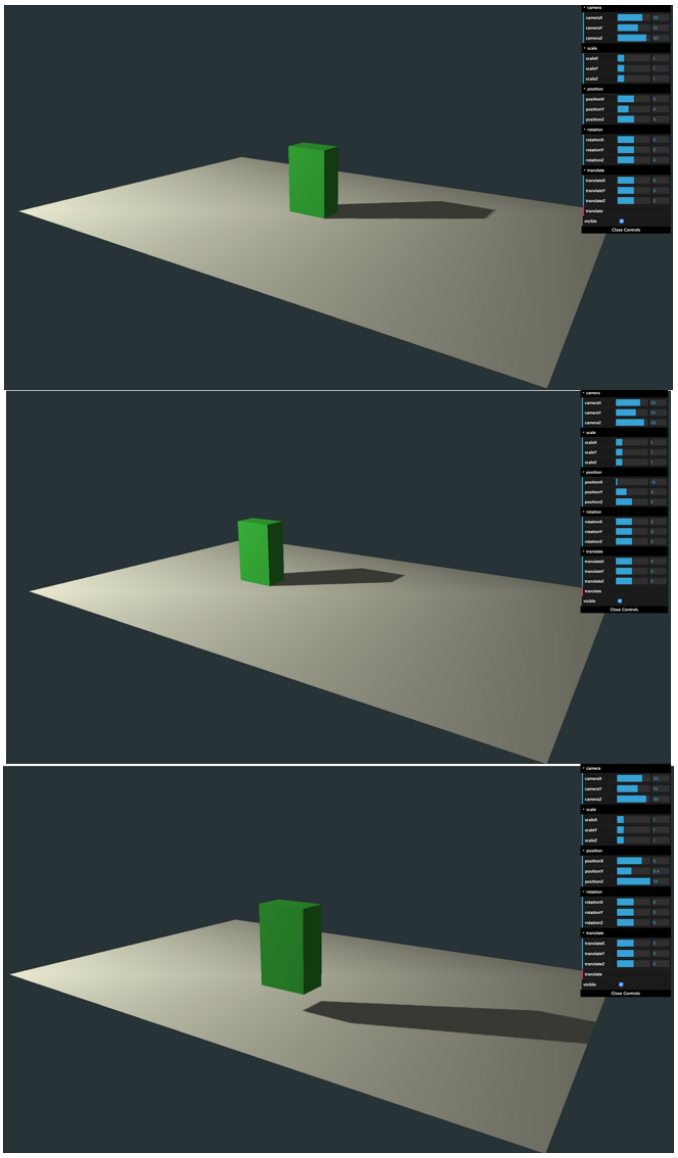

Fig. 22. 3D mapping JavaScript application.

IV. Conclusion

For this implementation for a sonar radar, the goal was to use as little as hardware as necessary while still fulfilling the requirements. The main pieces of hardware needed for this project were the ultrasonic sensor, a Zedboard, and servo motors. The sensors are capable of detecting objects from within 20 feet. Using the data from sensors we were then able to successfully calculate the necessary values through VHDL and SoC design. Once everything is calculated the data is able to be sent over to a remote server through a web socket, which is all done through the Linux OS which was installed into the Zedboard. This system created a completely wireless transfer of all of the data. Finally, the remote server then displays everything onto the radar interface.

The hope for the future is to replace the sonar sensors with the image sensors. Two image sensors would be able to calculate the distance and make it 3D.

References

[1] Ihara, Ikuo. "Ultrasonic Sensing: Fundamentals and Its Applications to Nondestructive Evaluation." Lecture Notes Electrical Engineering Sensors, pp. 287-305., doi:10.1007/978-3-540-69033-7_14.

[2] Marioli, D., et al. "Ultrasonic Distance Measurement for Linear and Angular Position Control." IEEE Transactions on Instrumentation and Measurement, vol. 37, no. 4, 1988, pp. 578-581., doi:10.1109/19.9817.

[3] M. Parrilla, J. J. Anaya, C. Fritsch, "Digital signal processing techniques for high accuracy ultrasonic range measurements", IEEE Trans. Instrum. Meas., vol. 40, pp. 759-763, Aug. 1991.

[4] D. Marioli, C. Narduzzi, C. Offelli, D. Petri, E. Sardini, A. Taroni, "Digital time-of-flight measurement for ultrasonic sensors", IEEE Trans. Instrum. Meas., vol. 41, pp. 93-97, Feb. 1992.

[5] A. Carullo, F. Ferraris, S. Graziani, U. Grimaldi, M. Parvis, "Ultrasonic distance sensor improvement using a two-level neural network", IEEE Trans. Instrum. Meas., vol. 45, pp. 677-682, April 1996.

[6] MaxSonar. LV-MaxSonar® -EZ™ Series High Performance Sonar Range Finder https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf

[7] Gillison, I. Leifer, USDA Agricultural Research Service, Variety Fabrication, Indiana University, Duke University - Department of Mechanical Engineering, Washlink Systems, Saskatchewan Research Council, and Elbit Systems Ltd., "Ultrasonic Sensors - Object Detection," Senix Ultrasonic Sensors. [Online]. Available: https://senix.com/object-detection/.

[8] "Ultrasonic Range Finder," ArcBotics - Ultrasonic Range Finder. [Online]. http://arcbotics.com/products/sparki/parts/ultrasonic-range-finder/.

[9] "Pmod MAXSONAR: Maxbotix Ultrasonic Range Finder," Digilent. [Online]. Available: https://store.digilentinc.com/pmodmaxsonar-maxbotix-ultrasonic-range-finder/.