Performance Analysis of PING))) Ultrasonic Distance Sensor

Elaine Cole, elainemcole (at) wustl.edu (A paper written under the guidance of Prof. Raj Jain)

Download 

Abstract:

The concept of a robot refers to a machine or device with the capacity to perform complex actions [MerriamWebster01]. In this current era of technology, robotics serves as more than just a branch of science [Wikipedia02], but a field with opportunity to improve the lives of people around the world. From designing a self-driving wheelchair for those in retirement communities struggling with mobility [Teo17] to building a robotic system to monitor neurons with a success rate comparable to manual performance of highly trained scientists [Trafton17], roboticists have made massive leaps in both the mechanical and electrical components of a robot. Today, robotic kits can be purchased for all ages, but as the use of robots in various industries increases, so does the need for precision in robotic systems; delay and error are unacceptable in certain real-time systems.

The objective of this study is to measure and analyze the precision of a particular home-built robot's ability to recognize objects using a Parallax PING))) Ultrasonic Distance Sensor (PING))) sensor) with a 2^k*r factorial design, beginning with an overview of key methods and terms used in this study's system and experiment, subsequently detailing the performance evaluation, and finally explaining the results and analyzing what they mean as it relates to the precision of the robot. The results show that the PING))) sensor for the given factors and workloads has an error large enough that, while this may be acceptable for a hobby robot project, this robotic system should not be used in the real world for measuring distance in critical situations.

Keywords: Experimental Design, Performance Evaluation, Performance Analysis, Arduino, Arduino Uno, Robot, PING))), Ultrasonic Distance Sensor, Parallax, BOE, Board of Education Shield, Boe-Bot, microcontroller

Table of Contents:

- 1. Introduction
- 2. System Characteristics
  - 2.1 PING))) Sensor
  - 2.2 BOE Shield-Bot Architecture
  - 2.3 Arduino Uno
  - 2.4 Software Design
- 3. Performance Evaluation
  - 3.1 System and Services

    - 3.2 Goal and Metrics
    - 3.3 Parameters and Factors
    - 3.4 Workloads
    - 3.5 Evaluation Technique
- 4. Results and Analysis
    - 4.1 Computation of Effects
    - 4.2 Estimation of Experimental Errors
    - 4.3 Analysis and Future Work
- 5. Summary
- 6. References
- 7. List of Acronyms

1. Introduction

The topic of robotics is not necessarily small; as stated previously, the term "robot" refers to a mechanism that can perform complex actions [MerriamWebster01]. Notably, this definition can include many concepts; what constitutes as a machine or device? What is required to be considered a "complex action"?

To successfully conduct a case study regarding robotics, we must first narrow our scope; rather than examining "robots" as a whole, we instead will measure and analyze a specific robotic system's process of sensing an object or obstruction within a set distance and area and transmitting that information from the sensor to the microcontroller and then to the roboticist. Using the data from our system evaluation, we will then draw conclusions regarding the precision of the robot's capability to acknowledge an obstruction with the intent of making an appropriate decision as to the subsequent movement path of itself.

In this paper, section 2 covers the characteristics of this system, including the PING))) sensor used, the hardware architecture of the robot, the Arduino board and microcontroller used, and the software built and compiled onto the Arduino board. Following this, we will outline the experimental design implemented, including the workload, goals, and factors analyzed. Section 4 presents the outcomes of the study after the measurements. We conclude with a summary containing final thoughts on the precision of this robot's sensor processing. References and a list of acronyms used in this study are included thereafter.

2. System Characteristics

This section provides an overview of the relevant technology and terms used in this system and experiment. First describing the uses, parts, and functionality of a PING))) sensor, we then include an explanation of the structure of the robotic system and architecture built for this experiment. Additionally, we discuss the Arduino Uno and its task within this robotic system and finally conclude this section with a summary of the code designed for and implemented by this robot.

2.1 PING))) Sensor

Below we discuss the features, uses, and capabilities of the PING))) sensor in relation to this robotic system.

A PING))) sensor measures the distance between itself and an object within a range of 2 cm and 3 m [Parallax04] using ultrasonic bursts. Used in many robotic applications, additional implementation of this sensor may include security systems as a form of detection. To measure distance, the PING))) sensor consists of the following:

- Emitter
- Detector
- Burst indicator light-emitting diode (LED)
- Male 3-pin header

The sensor's emitter sends out 40 kHz bursts, and the detector is an ultrasonic microphone which listens for the sound bursts to return. Depending on the "echo" of the burst, the sensor determines if there is an obstacle within range and where. The burst indicator LED turns on when the sensor is active, and the male 3-pin header consists of ground, a 5 Voltage Direct Current (VDC) power supply, and one bidirectional pulse interface input/output (I/O) pin [Arduino02].

To function with power and collect its measured distances, the PING))) sensor connects to a board and microcontroller with the male 3-pin header. Upon initialization, the microcontroller sends out a "start pulse," indicating to the PING))) sensor to begin collecting data, after which the sensor sends echo time pulses back to the microcontroller [Parallax04].

In Figure 1, a schematic of a PING))) sensor, the voltage drain (Vdd) refers to the 5 VDC power supply, the voltage source (Vss) is Ground (GND), and the I/O pin serves as the signal (SIG). The emitter, detector, and indicator LED are also labeled.

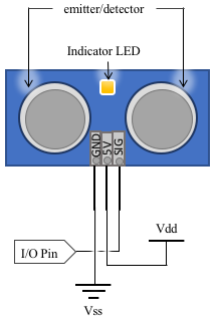

Figure 1: Schematic of a PING))) Sensor with pin labels.

Specifically, The PING))) sensor functions by broadcasting a chirp, or short ultrasonic 40 kHz burst, through its emitter and simultaneously administering a high output pulse into the I/O pin to the microprocessor, letting it know to begin tracking the echo time of the ultrasonic chirp. Once the detector is hit by the reflected ultrasonic chirp, the PING))) sensor reverts the output pulse to low on the I/O pin [Parallax04]. Figure 2 illustrates this process. The start pulse as labeled in Figure 2 is the initialization signal sent to the PING))) sensor by the microprocessor, telling it to

begin measuring. We will refer to the board including the microcontroller itself as the microcontroller, as the board only serves to assist the microcontroller.

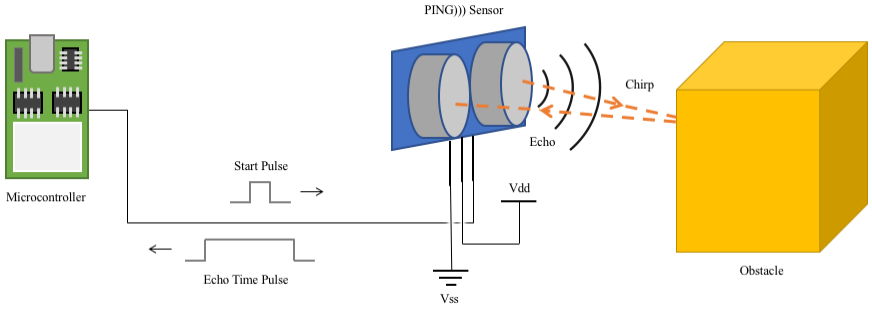

Figure 2: A PING))) sensor process flow detecting object and notifying microcontroller.

Having discussed the PING))) sensor and its ability to measure the distance an object is away from the sensor, we now proceed to discuss the architecture of the robot, modeled by a Board of Education (BOE) Shield-Bot [Parallax01], and how the PING))) sensor is attached.

2.2 BOE Shield-Bot Architecture

This section overviews the architecture of the BOE Shield-Bot, the basic frame of the robotic system built for this experiment.

The BOE Shield-Bot design comes from a movement driven by Parallax with the goal of encouraging others to get involved in robotics, specifically do-it-yourself projects. The BOE Shield-Bot is, in essence, a small microcontroller development platform [Lindsay16]. The kit itself includes a metal chassis, servo motors, wheels, the BOE Shield, and a battery pack along with other hardware such as screws and bolts [Parallax05].

This robotic system functions by connecting a microcontroller, discussed in the following section, to the BOE Shield, which connects to the servo motors and PING))) sensor. The microcontroller used here is an Arduino Uno, which tells the servo motors to spin, thus pivoting the wheels and moving the wheels forward, and the PING))) sensor to begin detecting through pins directly connecting the Arduino Uno to the BOE Shield. The BOE Shield, in turn, passes on the messages through wires to the servo motors and PING))) sensor [Lindsay16].

The PING))) sensor itself is attached by the PING))) Mounting Bracket Kit [Parallax04], and following the directions on how to attach the mounting bracket with screws yields a functional BOE Shield-Bot with the ability to rove around a space [Lindsay10]. Notably, however, in this experiment we are not concerned with the wheels and movement of this robotic system, but with the PING))) sensor and the accuracy of the data it collects.

Figure 3 presents a side view of the robot built for this experiment. As visible, the chassis holds together the wheels and mounting bracket kit holding the PING))) sensor and its connected servo motor, allowing it to adjust its viewpoint for measuring up to 180 degrees to the front

[Lindsay16]. The servo motors which control the wheels and battery pack are not easily seen in this figure.

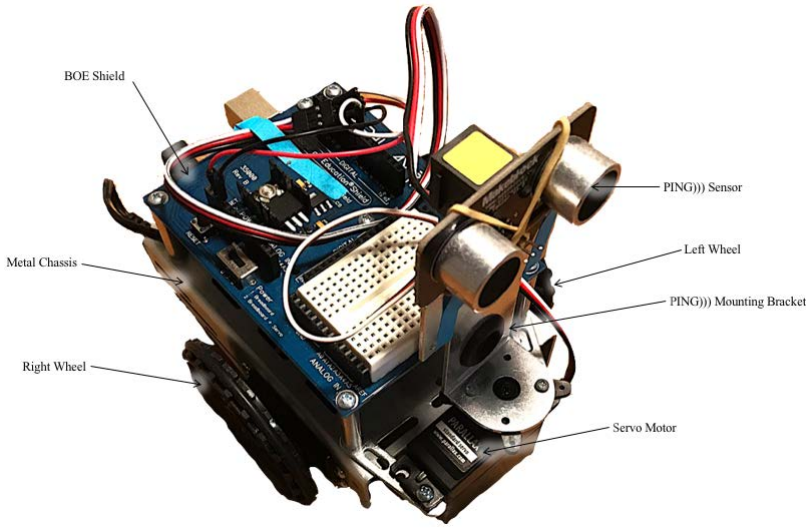

Figure 3: Side image of the final robotic system.

Additionally, Figure 3 shows us the BOE Shield, which connects directly to the Arduino Uno (underneath) with 22 American Wire Gauge (AWG) wires used. This BOE Shield serves as a connector to the "brain" of the robot: it's microcontroller, the Arduino Uno [Arduino01]. We will further discuss the Arduino Uno board and its relation to the BOE Shield in the following section.

2.3 Arduino Uno

In this section, we will discuss a brief history of Arduino and Parallax's own microcontroller, the integration of the Arduino Uno into the BOE Shield-Bot, and the function of the Arduino Uno as the brain of this robot.

Arduino itself serves as an open-source electronics platform with the goal of creating and using easy-to-use hardware and software [Arduino01]. To use an Arduino board, the only tools required (other than knowledge of the system) are the Arduino Software Integrated Development Environment (IDE), which serves as a platform to write Arduino-language code in and upload to an Arduino board with ease, and the board itself.

Notably, the guide followed for this project to build the BOE Shield-Bot for Arduino is actually a variation of the original Boe-bot robot, which included a different microcontroller than the Arduino Uno [Lindsay16]. The original design used a Beginner's All-purpose Symbolic Instruction Code (BASIC) Stamp 2 microcontroller [Parallax06], whose features include a Central Processing Unit (CPU) and built-in Read-Only Memory (ROM). Introduced in 1999, it used Parallax BASIC (PBASIC), a programming language intended for first-time software programmers. In 2005, Arduino released its own microcontroller, the Arduino Uno, which gained much popularity for its ability to build simple robotic systems at home [Lindsay16]. As a

result, Parallax worked with SimplyTronics to build the BOE Shield, a connection to allow the Arduino hardware to be compatible with the BOE-Bot chassis.

The Arduino Uno contains many components, but for our purposes we outline only the ones used in this experiment. These include a Universal Serial Bus (USB) port, a supply voltage port, digital and analog pins, a reset button, and a main integrated circuit [Arduino01]. The USB port serves to connect the Arduino Uno to a computer in order, for instance, for the roboticist to track the measurements in a datasheet, and for the roboticist to load code onto the board. The supply voltage port gives the Arduino Uno power, ideally between 6 and 12 V [Arduino01], which is the actual microcontroller itself. Analog pins are for sensors, such as the PING))) sensor, whereas digital pins are for "on or off" buttons and LEDs. The main integrated circuit contains the ROM and CPU of the Arduino Uno. Figure 4 depicts these parts of the board.

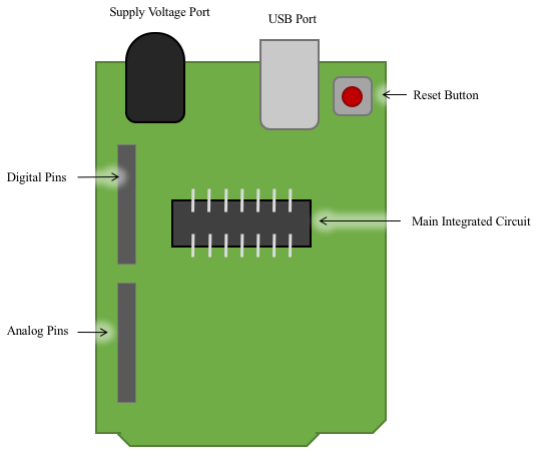

Figure 4: Primary components of Arduino Uno.

In conjunction with the BOE Shield-Bot, the Arduino Uno has the ability to execute a number of tasks, but specifically [Lindsay16]:

- Monitor the PING))) sensor to detect objects around the BOE Shield-Bot
- Form calculated decisions given the measurements received from the sensor
- Control its own motion using the servo motors and wheels
- Relay information (i.e., measurements from the PING))) sensor) to the roboticist

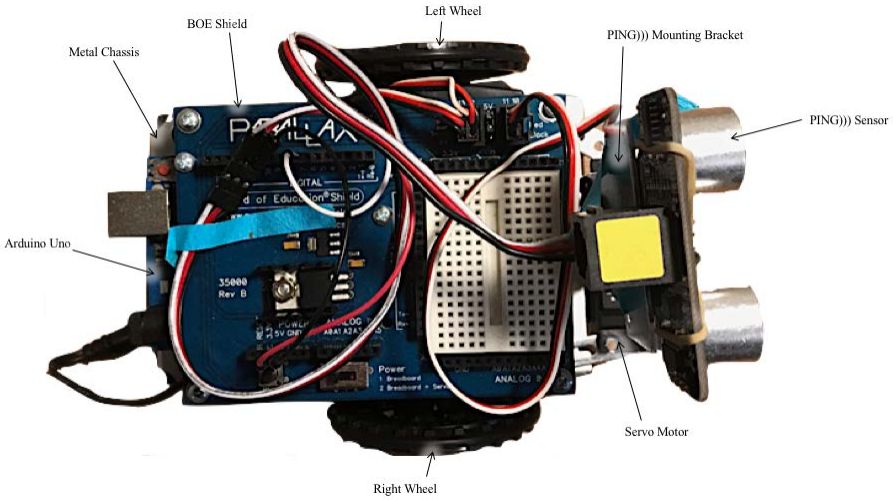

Figure 5: Top image of the final robotic system.

Figure 5 contains an additional image of the robotic system, top-view. Here, the Arduino Uno can be seen underneath the BOE Shield, with its USB port visible. Now that we have successfully built the hardware-aspect of the robot, we may write and upload code to the Arduino Uno board. But having the physical hardware, albeit connected properly, is not sufficient for a successful, deployable robot. We will now discuss the software design implemented for this experiment.

2.4 Software Design

In this section, we will discuss the software design of the code used to implement this robot's functionality, including the basics of Arduino programming language, the skeleton supplied for the code built for this study, and the changes made to develop the final version of said code.

Arduino programming language is based on Wiring, an open-source programming framework specifically designed for use with microcontrollers [Wiring01]. The syntax and form itself, however, has a likeness to that of C++, an object-oriented programming language designed by Bjarne Stroustrup in 1979 [CPlusPlus01]. The Arduino programming language can be divided into three primary sections [Arduino03]:

- Structures
- Values
- Functions

In this case, the structure refers to elements of the Arduino most like the C++ language, such as arithmetic, comparison, and bitwise operators [Arduino03]. The values include variables and constants, and the functions are used for controlling the board and completing computations [Arduino03]. While overall code varies greatly, Arduino code contains a "setup" function, which is run upon initialization, and a "loop" function, which is subsequently run repeatedly [Lindsay16].

The code built for this study was originally forked from a tutorial by Parallax as an open-source project in 2012 [Parallax01] to be used for a BOE Shield-Bot application that roams with Ping))) sensor mounted on servo turret [Cole17]. Later revisions and the final version have been committed to a separate GitHub repository found at [Cole17].

The full code found in this repository cited above functions to maneuver the robot around objects that it senses, relaying measured echo time taken from the PING))) sensor back to the roboticist and making a decision as to where to move to next [Parallax02]. For this study, however, the Arduino is not given power to its servo motors and as such does not move because we are only studying the measurements taken from the PING))) sensor, and not the time the robot needs to adjust its path.

We have now discussed the architecture of the hardware for the robot used in this study, including the PING))) sensor used to detect the distance an object is away, the BOE Shield-Bot hardware, and the Arduino Uno microcontroller used as the brain of the robot, as well as the software uploaded to the Arduino board to give it commands and send the measurements collected back to the roboticist. We will now discuss the experimental design implemented, including the workload, goals, and factors analyzed in this study, in section 3.

3. Performance Evaluation

This section describes the components of this measurement study, including the definition of the system and services, the goal and metrics, the prevalent parameters and factors, and the evaluation technique used. Finally, we discuss the experimental design. This process of a systematic approach to performance evaluation is taken from [Jain91].

3.1 System and Services

An important part of the first step of the systematic approach to performance evaluation is to define the system through delineating system boundaries [Jain91]. Here we define the system as including the microcontroller, PING))) sensor [Parallax02], chassis and all connections between these system aspects, the computer connected to the microcontroller to collect and track the measurements, and any object within the range of 2 cm to 3 m in front of the PING))) sensor [Arduino02]. Notably, we do not need to include the servo motors or wheels in this system as we are not measuring or analyzing the movement of the robot.

The services of this system, that is, what the system provides [Jain91], are to measure the round-trip echo time of an ultrasonic sound wave administered by the PING))) sensor and to transmit the measurements to the roboticist through the computer.

The implementation of this system is an Arduino Uno as the microprocessor, a BOE Shield to connect the PING))) sensor to the Arduino Uno, and a computer. Having defined the system of this study and the services provided, we next state the goal and metrics, following the systematic approach to performance evaluation [Jain91].

3.2 Goal and Metrics

Another key component of the first step in the systematic approach to performance evaluation is to state the goals for the study [Jain91]. In this case, as detailed previously, the goal is to measure and analyze this robotic system's procedure of sensing an obstruction within the set distance and transmitting that information from the sensor to the microcontroller and then to the roboticist. The intent is that we will be able to draw conclusions regarding the precision of the robot's capability to acknowledge an obstruction with the goal of making an appropriate decision as to the subsequent movement path of itself.

The next step is to select metrics, which are the criteria selected to compare the performance [Jain91]. In the case of this study, our metrics include the accuracy of the distance measured by the PING))) sensor that the robot is from an object within 2 cm and 3 m in front of it and calculated from microseconds to centimeters by the code.

Continuing with the systematic approach, we will now define parameters and factors.

3.3 Parameters and Factors

We will now list the parameters and factors of the study. The list of parameters is not exhaustive, and can be divided into system parameters and workload parameters [Jain91]. The system parameters generally do not vary by installation and include both hardware and software parameters, whereas the workload parameters, characteristics of the users' requests, do vary by installation [Jain91]. The following Table 1 lists system parameters that may have an effect on the accuracy of the measured distance.

Table 1: System Parameters

Parameters	Description
Arduino Board	Arduino Uno Revision 3
Microcontroller in Arduino Boards	ATmega328
Computer	2016 MacBook Pro Laptop
Computer Operating System	MacOS High Sierra Version 10.13.1
PING))) Sensor	Parallax Product #28015, "PING))) Ultrasonic Distance Sensor"
Connecting Board	Parallax Product #35000, "Board of Education Shield (for Arduino)"
Wires Type Used	22 AWG Wire
Connection between Arduino Board and Computer	USB Cable

Here, the Arduino Uno Revision 3 includes the ATmega328 processor which allows the user to upload new code via a bootloader, rather than needing external hardware programming skills. Table 2 lists the workload parameters that may affect the measured distance, including configurable variants.

Table 2: Workload Parameters

Parameters	Description
Distance between object and robot	Configurable
Set minimum distance away to notify user	Configurable

We list the factors, that is, parameters that we choose to vary [Jain91], and their levels, or values, in Table 3. These parameters deal with different distances of a set object and the robotic system and the minimum distance in which the robotic system will notify the roboticist of the object, which are most likely to have a direct impact on determining the accuracy of the measured distance.

Table 3: Factors under test

Parameters	Description
Distance between object and robot	5 cm/50 cm
Set minimum distance away to notify user	10 cm/40 cm

Due to limitations in resources, all other parameters are fixed. Upon careful listing of our parameters and factors, we continue in the systematic approach to performance evaluation to discuss the workload of the system.

3.4 Workloads

Following the systematic approach, we will detail the workload, a list of service requests to the system [Jain91]. In this study, the workloads consist of 30 second intervals of the PING))) sensor moving to 5 different servo turret (that is, the servo motor controlling the PING))) sensor) positions [Cole17] as listed below in Table 4. During these 30 sec intervals of measurement, the servo turret will alternate between the 5 different turret positions every 0.1 sec [Cole17].

Table 4: Array indices of servo turret positions

Array Index	Degree Value
0	180
1	135
2	90
3	45
4	0

The intent with this workload is to provide sufficient time and range for the PING))) sensor to detect an object within 180 degrees in front of the robotic system, within a range of 2 cm and 3 m. According to the datasheet for the PING))) sensor used in this system implementation, there are 73.746 microseconds per inch, given that sound travels at 1130 feet per second [Parallax04]. In the code, we round this to 74, and then divide by 2, as the distance traveled by the ultrasonic "ping" includes the outbound as well as the return [Cole17].

Knowing this workload, we will now discuss the evaluation technique used in this study. This will conclude our systemic approach to performance evaluation, excluding results and analysis [Jain91].

3.5 Evaluation Technique

Having identified our workload and defined the overall system to be studied, we now describe the evaluation technique used to assess said system. We review the primary techniques for performance evaluation, choose the best option for our system, and define the design for the selected technique.

The three basic techniques to performance evaluation are as follows:

- analytical modeling
- simulation
- measurement of a real system

Analytical modeling may include mathematical modeling, simulation commonly involves the creation of a digital prototype to mimic a real system, and measuring a real system requires the actual system to be built [Jain91]. When selecting an evaluation technique, a primary concern is the lifecycle of the system [Jain91]: that is, whether or not the system has been created yet or not. In the case of our defined robotic system, we have built a functional prototype [Cole17] and thus are not limited to analytical modeling and simulation had we not yet fabricated it. Other criterion considered include the time required, resources and tools available, the level of accuracy needed, the cost to construct the system, and the saleability of results [Jain91].

Having reviewed these considerations, the measurement of a real system is the best option for this evaluation primarily given the resources available to build the system and knowledge of robotics and the saleability of having demonstrated and experimented with the real system. Notably, a major rule of validation is to check the results of a measurement by simulation or analytical modeling [Jain91], but given the duration of time available, this experiment may only include one evaluation process and design.

Having determined that we will be measuring a real system, we now choose our design. Specifically, we will be using a 2^k*r factorial design to describe this study, where k, the number of factors, is 2, and r, the number of repetitions, is 6. This allows for 2^k*r observations, and estimation of experimental errors [Jain91]. The 2 factors and their levels are detailed in Table 5.

Table 5: factor symbols and levels

Symbols	Parameters	Level –1	Level +1
A	Distance between object and robot	5 cm	50 cm
B	Set minimum distance away to notify user	10 cm	40 cm

For each combination of factors, measurements are taken with the workloads. The object used as an obstruction is a 6 cm wide tennis ball, placed at the different factor levels along a straight line directly in front of the robot and its PING))) sensor at turret position 2. Figure 7 shows the basic equation for a 2^2*r factorial design, where e is the experimental error and the q's are the effects of each factor [Jain91].

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B + e$$

Figure 7: Mathematical model for a 2^2*r factorial design

We have defined our evaluation technique and design as a real-system measurement of a 2^2*6 factorial design. We now conduct the experiment and will analyze our results in the following section.

4. Results and Analysis

In this section, we will analyze our results from our experimental design outlined in the previous section. Specifically, we compute the effects of our 2 identified factors, the distance between the object and the robot and the minimum distance at which we direct the robotic system to notify the roboticist of the obstacle, calculate the experimental errors, and interpret our data and suggest next steps for continued analysis of the accuracy of the system.

4.1 Computation of Effects

This section discusses the computation of effects of the 2^k*r factorial design for k = 2 and r = 6. To analyze a 2^2*r design, the easiest way is to use a sign table [Jain91] as shown in Table 6 below. When y = 1, an alert was made to the roboticist that an object was recognized. When y = 0, an alert was not made.

Table 6: Sign table of the 2^2*6 design

I	A	B	AB	y	Mean ȳ
1	–1	–1	1	(1, 1, 1, 1, 1, 1)	1
1	1	–1	–1	(0, 0, 0, 0, 0, 0)	0
1	–1	1	–1	(0, 0, 1, 0, 1, 1)	0.5
1	1	1	1	(0, 0, 0, 0, 0, 0)	0
1.5	**–1.5**	**–0.5**	**0.5**		**Total**
0.125	**–0.375**	**–0.125**	**0.125**		**Total / 4**

Here, given that r = 6, we sum the individual observations under y for each experiment and divide by 6 to get the sample mean of y. The values in each of the first four columns are the multiplied by those in the last column, sample mean of y, and is the "total" for each column. Then, we divide by 4 to yield the following [Jain91] in Figure 8:

$$q_0 = 0.125$$
$$q_A = -0.375$$
$$q_B = -0.125$$
$$q_{AB} = 0.125$$

Figure 8: Computed effects for the 2^2*r design

Now that we have computed the effects for the given 2^k*r factorial design, we will calculate the errors of each factor value.

4.2 Estimation of Experimental Errors

In this section, we use the mathematical model for a 2^2*r design given in Figure 7 to estimate the response for the factor values [Jain91]. Specifically, we adjust the model to show the following Figure 9:

$$\hat{y}_i = q_0 + q_A x_{Ai} + q_B x_{Bi} + q_{AB} x_{Ai} x_{Bi}$$

Figure 9: Model for estimation of errors of 2^2*r design

In this equation, the y-value is the estimated response when the factors A and B are at the given x-value levels [Jain91]. Table 7 shows the computation of errors for the 2^2*r design. The experimental error e, is the difference in the measured y and estimated y per observation.

Table 7: Computation of errors for 2^2*r design

	Effect				Estimated Response	Measured Responses						Errors					
	I	A	B	AB													
i	0.125	−0.375	−0.125	0.125	$\hat{y}_i$	y_{i1}	y_{i2}	y_{i3}	y_{i4}	y_{i5}	y_{i6}	e_{i1}	e_{i2}	e_{i3}	e_{i4}	e_{i5}	e_{i6}
1	1	−1	−1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
2	1	1	−1	−1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	−1	1	−1	0.75	0	0	1	0	1	1	−0.75	−0.75	0.25	−0.75	0.25	0.25
4	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Now that we have calculated the errors in each experiment, we will interpret our data in the following section. Additionally, we will provide insight into future possible work to further our conclusions from this experiment.

4.3 Analysis and Future Work

In this section, we interpret our data outlined in the previous section. Then, we suggest future work of expanding this experiment to further this study.

Given the computed errors for the experimental design, we can see that experiment 3 had errors, while experiments 1, 2, and 4 proceeded as expected. It is possible, given the concern for environmental errors in measuring a real system as the experimental technique [Jain91], that there was an issue in experiment 3 outside of our determined factors, such as an accidental hand movement in the way of the PING))) sensor, that affected our results. Further data will be needed to confirm this, as, while experiment 3's data might be considered an outlier, it is possible it could be an issue with the system.

In a real-world situation, a 25% error would not be acceptable. One can argue that it is acceptable for this particular robotic system, given that it is a home-built robot with no responsibility or effect, but, if this were to be put to work in a critical environment, we would not recommend relying on the data to be precise given the results of this experiment.

Future work might consist of more levels of our 2 identified factors. Additionally, expanding the factors to include the time at which the servo motor, or turret, controlling the PING))) sensor's angled direction. It is possible that the 0.1 sec that the PING))) sensor has at each of the 5 set positions is not enough, even though the PING))) sensor's echo takes milliseconds. Expanding the experiment to include extending the time interval at each position will give us a better idea of it's effect on the accuracy of the robotic system.

5. Summary

Having interpreted our data, we now summarize what we've discussed and taken away from this study, working chronologically through the paper.

In this study, we discussed the prevalence of robotics in the present day. We introduced the topic of robotics as a complex concept and narrowed the scope of our study. We then outlined the characteristics of our system for our experiment, including the hardware involved such as PING))) sensor, BOE Shield-Bot architecture, and Arduino Uno, as well as the software built for this robotic system.

We then proceeded to discuss the performance evaluation using the systematic approach, outlining the system, services, goal and metrics of the system, the parameters and identified factors, and the workloads. We then discussed our evaluation technique, a 2^k*r factorial design, and then took measurements.

Upon discussion and analysis of our results, it is apparent that, while the accuracy of this robotic system in detecting obstacles at various distances from its PING))) sensor may be sufficient for small home projects, in a real-world situation in which precision is vital, this system is not sufficient.

6. References

- [Arduino01] "Arduino - Introduction," https://www.arduino.cc/en/Guide/Introduction. This introduces Arduino and its potential applications.
- [Arduino02] "Arduino - Ping," https://www.arduino.cc/en/Tutorial/Ping. This provides an overview to the ultrasonic range sensor and its hardware setup.
- [Arduino03] "Arduino Reference," https://www.arduino.cc/reference/en/. This is a language reference for Arduino, including structures, values, and functions.
- [Cole17] "elainecole/columbus: Arduino-based robotic explorer," https://github.com/elainecole/columbus. The GitHub repository containing code used for the robot in this experiment.
- [CPlusPlus01] "A Brief Description - C++ Information," http://www.cplusplus.com/info/description/. This web page outlines a basic description of C++ programming language.
- [Jain91] R. Jain, "The Art of Computer Systems Performance Analysis," John Wiley and Sons, Inc., 1991, 9780471503361.

- [Lindsay10] A. Lindsay, "Robotics with the Boe-Bot," Parallax Inc., 2010, 9781928982531. This guide serves as a series of step-by-step Boe-Bot projects to engage entry-level students with robotics.
- [Lindsay16] A. Lindsay, "Robotics with the Board of Education Shield for Arduino," Parallax Inc., 2016, 9781928982616. This tutorial aided in building the first version of the robot used in this experiment.
- [Martin05] J. Martin, J. Williams, K. Gracey, A. Alvarez, S. Lindsay, "BASIC Stamp Syntax and Reference Manual," Parallax Inc., 2005, 9781928982326. This manual serves as an introductory reference manual to BASIC Stamp commands, architecture of various models, and the editor.
- [MerriamWebster01] "Robot," https://www.merriam-webster.com/dictionary/robot. This web page includes the dictionary definition of "robot."
- [Parallax01] "Roaming with PING Arduino Code," https://learn.parallax.com/tutorials/robot/shield-bot/shield-bot-roaming-ping/roaming-ping-arduino-code. This code served as the skeleton for what later became the final code used for this experiment.
- [Parallax02] "PING))) Ultrasonic Distance Sensor," https://www.parallax.com/product/28015. This web page includes a product overview, buying information, and range details of the PING))) sensor.
- [Parallax03] "PING))) Mounting Bracket Kit," https://www.parallax.com/product/570-28015. This web page details the mounting bracket kit used in this experiment.
- [Parallax04] "PING))) Ultrasonic Distance Sensor (#28015)," https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf. This product guide details key functions of a PING))) sensor, features, and how it functions with example programs.
- [Parallax05] "Robotics Shield Kit (for Arduino)," https://www.parallax.com/product/130-35000. This web page includes product information for the shield kit for Arduino Uno.
- [Parallax06] "PBASIC Programming with the BASIC Stamp," https://learn.parallax.com/educators/teach/pbasic-programming-basic-stamp. This web page outlines the function of the BASIC Stamp 2 microcontroller and the definition of PBASIC.
- [Teo17] "Featured video: A self-driving wheelchair," http://news.mit.edu/2017/featured-video-self-driving-wheelchair-0726. This news article describes the self-driving wheelchair invented to aid those struggling with mobility.
- [Trafton17] "Robotic system monitors specific neurons," http://news.mit.edu/2017/robotic-system-monitors-specific-neurons-0830. This news article highlights the effectiveness of a robotic system made by MIT engineers which monitors neurons in a living brain.
- [Wikipedia01] "History of Robots," https://en.wikipedia.org/wiki/History_of_robots. This article outlines the history of robotics, spotlighting early beginnings.
- [Wikipedia02] "Robotics," https://en.wikipedia.org/wiki/Robotics. This article includes the definition of robotics, its history, applications, and primary components.
- [Wiring01] "Wiring," http://wiring.org.co/. This is the web site for Wiring, an open-source programming framework for microcontrollers.

7. List of Acronyms

- BASIC: Beginner's All-purpose Symbolic Instruction Code
- BOE: Board of Education
- CPU: Central Processing Unit
- GND: Ground
- I/O: Input/Output
- IDE: Integrated Development Environment
- LED: Light-Emitting Diode
- PING))) Sensor: PING))) Ultrasonic Distance Sensor
- SIG: Signal
- ROM: Read-Only Memory
- USB: Universal Serial Bus
- VDC: Voltage Direct Current
- Vdd: Voltage Drain
- Vss: Voltage Source

Last modified: December 15, 2017