NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DESIGN AND PROTOTYPING OF A PORTABLE NAVAL WEAPON CONTROL SYSTEM FOR HEAVY MACHINE GUNS

by

Evangelos Serris

June 2018

Thesis Advisor: Xiaoping Yun
Second Reader: James Calusdian

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE DESIGN AND PROTOTYPING OF A PORTABLE NAVAL WEAPON CONTROL SYSTEM FOR HEAVY MACHINE GUNS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Evangelos Serris	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Close-in weapon systems (CIWS) are an essential, computer-controlled defensive measure available on modern warships. Due to their size and weight, however, CIWS are deployed only on warships with a displacement over 1000 tons. Smaller ships, such as patrol boats, still carry and use less accurate crew-controlled heavy machine guns. We propose a small, light-weight, and portable CIWS that can be easily installed onto smaller warships. The proposed CIWS incorporates automated target tracking to reduce errors that arise from manual operation.

A prototype system, consisting of a stabilizing platform and a gun control unit, was designed and constructed for this research. The prototype incorporated sensors and microcontrollers to provide an automated target tracking capability. Two independent closed-loop controllers were implemented integrating inertial and vison-based sensors to provide automatic stabilization and tracking. A support structure for the prototype was fabricated using parts made with a three-dimensional printer.

Through experimental measurement and simulations, performance of the prototype was evaluated and compared favorably with that of existing CIWS systems; however, during the course of this work, we found that several improvements would be required to make the proposed portable CIWS a viable solution. The work highlighted the need for an upgraded inertial sensor, motor gear assemblies that have much less backlash, and stronger supporting structure.

14. SUBJECT TERMS close-in weapon system, PID controller, computer vision, Arduino microcontroller	**15. NUMBER OF PAGES** 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

DESIGN AND PROTOTYPING OF A PORTABLE NAVAL WEAPON CONTROL SYSTEM FOR HEAVY MACHINE GUNS

Evangelos Serris
Lieutenant, Greek Navy
B.S., Hellenic Naval Academy, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2018

Approved by: Xiaoping Yun
Advisor

James Calusdian
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Close-in weapon systems (CIWS) are an essential, computer-controlled defensive measure available on modern warships. Due to their size and weight, however, CIWS are deployed only on warships with a displacement over 1000 tons. Smaller ships, such as patrol boats, still carry and use less accurate crew-controlled heavy machine guns. We propose a small, light-weight, and portable CIWS that can be easily installed onto smaller warships. The proposed CIWS incorporates automated target tracking to reduce errors that arise from manual operation.

A prototype system, consisting of a stabilizing platform and a gun control unit, was designed and constructed for this research. The prototype incorporated sensors and microcontrollers to provide an automated target tracking capability. Two independent closed-loop controllers were implemented integrating inertial and vison-based sensors to provide automatic stabilization and tracking. A support structure for the prototype was fabricated using parts made with a three-dimensional printer.

Through experimental measurement and simulations, performance of the prototype was evaluated and compared favorably with that of existing CIWS systems; however, during the course of this work, we found that several improvements would be required to make the proposed portable CIWS a viable solution. The work highlighted the need for an upgraded inertial sensor, motor gear assemblies that have much less backlash, and stronger supporting structure.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

2D	Two-dimensional
3D	Three-dimensional
CIWS	Close-In Weapon Systems
CV	Computer Vision
DC	Direct Current
DEW	Directed-Energy Weapon
DOF	Degrees-of-Freedom
FLIR	Forward Looking Infrared
GMWS	Guided Missile Weapon System
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
LOS	Line-of-Sight
LOF	Line-of-Firing
LWS	Laser Weapon System
OTHT	Over-the-Horizon Target
PID	Proportional–Integral–Derivative
PWM	Pulse-Width Modulation
RAM	Rolling Airframe Missile
RPM	Rounds per Minute

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Xiaoping Yun, for his support and his guidance throughout this thesis. I would also like to give special thanks to Dr. James Calusdian who patiently encouraged my efforts and vigorously contributed in accomplishing by any means this idea. My special gratitude goes to the NPS Electrical Engineering chairman, Dr. R. Clark Robertson, and my academic advisor, Dr. Preetha Thulasiraman, for their substantial support during the last two years.

I am also grateful to my family for their constant motivation throughout my life. My father, Thomas, has always been my rock and the example to follow. My mother, Konstantina, taught me my first English words and always encourages me to follow my dreams. My dear friend, Dimitris, who by all means, showed me the way to NPS.

Lastly, I have to single out my beautiful wife, Morfoula, for a really special thank you. It wouldn't be the same without you. You make my life complete.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. NAVAL CLOSE-IN WEAPON SYSTEMS

Close-in weapon systems (CIWS) are the principal self-defense system of today's battleships. Nearly all warships having a displacement over 1000 tons are equipped with a CIWS [1]. It is the ship's last defense against incoming threats, such as missiles, rockets, and fast-moving craft. These systems consist of three major components: the sensor unit, the computer unit, and the gun control unit. Based on the type of firing unit, systems are classified as gun-based CIWS, Guided Missile Weapon Systems (GMWS), or Laser Weapon Systems (LWS). Their sensors are a combination of radars, Forward-Looking Infrared (FLIR) thermal cameras, and electro-optical sensors. The computer unit is the interface between the sensors and the gun control unit, with the operator having overall control of the system. CIWS are usually found on larger ship platforms because a typical CIWS is large and heavy, requiring dedicated space and ship's resources above and below deck [1]. On smaller ships with a displacement less than 1000 tons, crew-controlled heavy machine guns are used instead of the larger CIWS due to the lack of space. This defense system alternative introduces targeting and firing errors since it is manually operated, and the operator must anticipate the target's future position.

B. HISTORICAL REVIEW OF CIWS

Defense companies initially designed CIWS to defeat short-range air and surface threats. In the Cold War, the evolution of anti-surface missiles indicated the need for a new naval self-defense system. The Russian AK-630, developed between 1960 and 1970, is the ancestor of the modern CIWS [2]. The AK-630 is fully automatic and uses a six-barrel 30 mm machine gun. It combines a radar and an electro-optical sensor to track air and surface targets at a range of up to four kilometers [2]. Its rate of fire is 4,000 to 5,000 rounds per minute, and it is controlled from either its control console or a mounted gunsight. In 1979, the Russians developed the AK-630M version to resolve some of the major tracking and firing problems of the initial system [3]. Finally, in 2012, they developed the AK-630M-2 version after several modifications.

In the same time period, several other countries developed similar CIWS to upgrade their ships' air-defense capabilities. The Italian Dardo, the American Phalanx, and the Dutch Goalkeeper are some of the major CIWS that followed the AK-630 after 1980. Follow-on modifications improved the firing accuracy and tracking capabilities. The CIWS were also used as point-defense systems for several shore bases to protect valuable facilities and airports from missile threats [2]. Although the most recent gun-based systems can defend against several incoming threats, their effectiveness in many cases is poor. Projectile fragments, which emanate from the destroyed targets, have enormous kinetic energy and can cause severe damage to the defending ship. This security gap and the need for defensive measures over a larger area led to the development of GMWSs, such as the RIM-116 Rolling Airframe Missile (RAM). Today, the U.S. directed-energy weapon (DEW) or LWS marks the beginning of a new era for the naval CIWS.

In spite of the many improvements made to CIWSs over several decades of development, they are still large, heavy structures that are usually only available on larger warships. Their average weight is over five tons, and their dependence on the ship's power supply and sensors makes them incompatible with smaller warship platforms [1]; moreover, their total cost, as well as expensive maintenance, prohibits their installation on smaller platforms.

C. PORTABILITY IN NAVAL DESIGN

Earlier naval architecture utilized passive self-defense systems, such as heavily armored superstructures. Modern naval architecture focuses on active self-defense and portability [4]. With this design approach, modern ships are platforms consisting of components such as mechanical systems and weapon systems that adapt and change their mission quickly and easily. Modular warships with plug-and-play systems are regarded as the future of naval architecture. Yet, the design of CIWS does not adhere to this design philosophy or evolve in the same way. They are to this day large, over-weight systems that are hard-mounted to the ship's infrastructure; consequently, sailors cannot easily transfer and install them on smaller ship platforms.

In this thesis research, we investigate the feasibility of developing a CIWS suitable for smaller ships. The main requirements of a CIWS for smaller ships are that it be lightweight, modular, and portable. This allows for the CIWS to be easily installed on a warship of smaller size and be quickly adapted to a newly assigned mission or objective. A portable CIWS must be totally independent from the ship's sensors and power supply. Another desirable characteristic is for the CIWS to have a high degree of autonomy through the use of automated systems integrating sensors and actuators. This is required to accommodate the reduced crew size available on smaller ships and in naval systems proposed in the future.

D. THESIS OBJECTIVE

The modern CIWS is the point of the arrow in naval self-defense architecture. Such systems defend against all incoming threats and are the last defensive measure before impact. Due to their size and their dependence on a ship's sensors, these units are usually installed only on larger ship platforms. While defense companies around the world are trying to improve the effectiveness of these systems, designers have made no progress towards making them available for smaller ship platforms. In this thesis, we investigate the feasibility of developing a portable, easily transferable CIWS for installation on smaller warships.

There are a number of major requirements for a CIWS intended for smaller warships. First, with regard to the dimensions and the weight of the system, the CIWS must be compact and lightweight so that it can be easily moved and positioned as required by a limited crew with little or no support equipment; therefore, the prototype must be less than one cubic meter in size and less than 50 kg in weight. Although the typical firing range of a larger CIWS is 2,000 to 4,000 m, the firing range for the prototype does not need to exceed 1,500 m. This range is sufficient to defend against all incoming threats, such as fast patrol boats and fast-flying objects. Nonetheless, its firing accuracy must be comparable to that of existing systems. Finally, we limit the system's power source to a 24-V/20-A battery source that can be easily installed.

E. THESIS OUTLINE

As described in this chapter, the objective of this thesis is to research the design and construction of a portable CIWS for smaller ship platforms. In the remaining chapters of this thesis, we detail the research as follows. In Chapter II, we provide some background information about the accuracy of the existing systems, and we analyze the natural motion of a typical patrol boat. We conclude Chapter II with a brief description of the proposed design to familiarize the reader with the prototype. In Chapter III, we introduce the hardware and describe the operation of the components used. In the first part of Chapter IV, we describe the experimental measurements that were conducted to measure the performance of the prototype. Through analysis of the measured performance, we report on the accuracy of the prototype and compare it with that of existing CIWS systems. In the second part of Chapter IV, we use a simulation to extrapolate the expected performance of the prototype CIWS for dynamic inputs that cannot be easily reproduced experimentally. In Chapter V we summarize the thesis research and conclude with recommendations for future work.

II. BACKGROUND

In this chapter we provide some useful background before discussing the CIWS prototype. We describe the method used to estimate the CIWS's firing accuracy for a typical patrol boat. This estimate is used later in the thesis to evaluate the performance of the prototype. In this chapter, we also present the pitch and roll measurements made aboard a ship to quantify the dynamic environment for the CIWS prototype. Lastly, a brief description of the design approach is presented.

A. CIWS ACCURACY

In this section, we develop a method to estimate the firing accuracy of an existing CIWS. This is done to provide a baseline level of performance for comparison with our proposed prototype discussed later in the thesis. The method presented here makes use of publicly available video images that were taken during the operation of a typical CIWS. These videos are available online on YouTube; they were recorded during several counterterrorist operations. An example of a typical video examined to estimate CIWS accuracy is shown in Figure 1. While this method is not necessarily very accurate, it nonetheless provides a rough estimate of CIWS's firing accuracy.

Figure 1. CIWS Phalanx Operating against Inflated Boat. Adapted from [5].

The Line-of-Sight (LOS) is the imaginary line drawn between the gun control unit and the target, as shown in Figure 2. The Line-of-Fire (LOF) corresponds to the trajectory of the round fired by the gun control unit and is aligned with the gun's barrel. Ideally, the LOF is the imaginary line that connects the gun control unit with the target's future position. Stated in another way, the LOS is associated with the target's current position and the LOF with the future position. When the target's relative velocity with respect to the gun control unit is zero, the LOS is coincident with the LOF. During successful tracking of a target, a firing system estimates the LOS. By observing the target's relative motion, the firing system calculates the target's future position and computes the LOF. In this thesis, we only examine the tracking response of the prototype and estimated the system's tracking angular accuracy over the LOS.

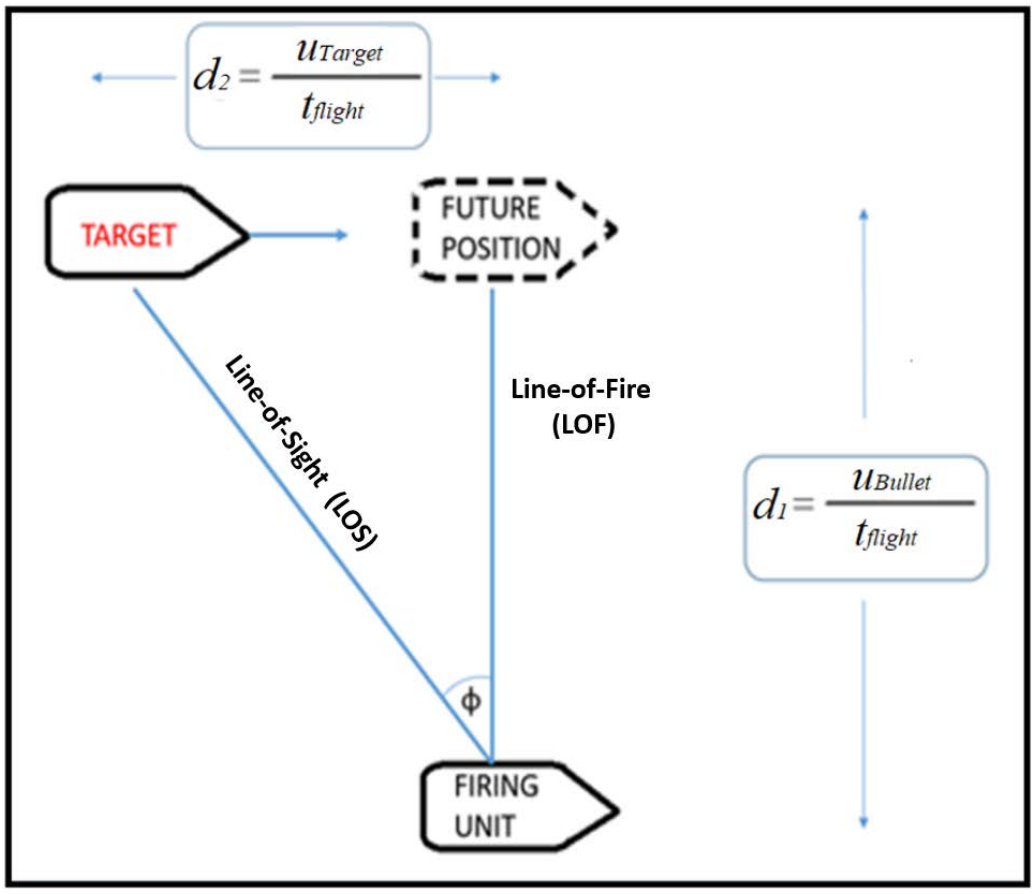

Figure 2. Line-of-Sight and Line-of-Fire Representation

When a CIWS is firing, the impact zone R_{IZ} around the target defines an area of error, as seen in Figure 3. The firing distance R_{FD} is the range between the firing unit and the target. Through a detailed examination of the available video frames, we roughly estimated both R_{IZ} and R_{FD}. These values were then used to estimate the angular accuracy θ_s using

$$\theta_s = \arctan\left(\frac{R_{IZ}}{R_{FD}}\right). \tag{1}$$

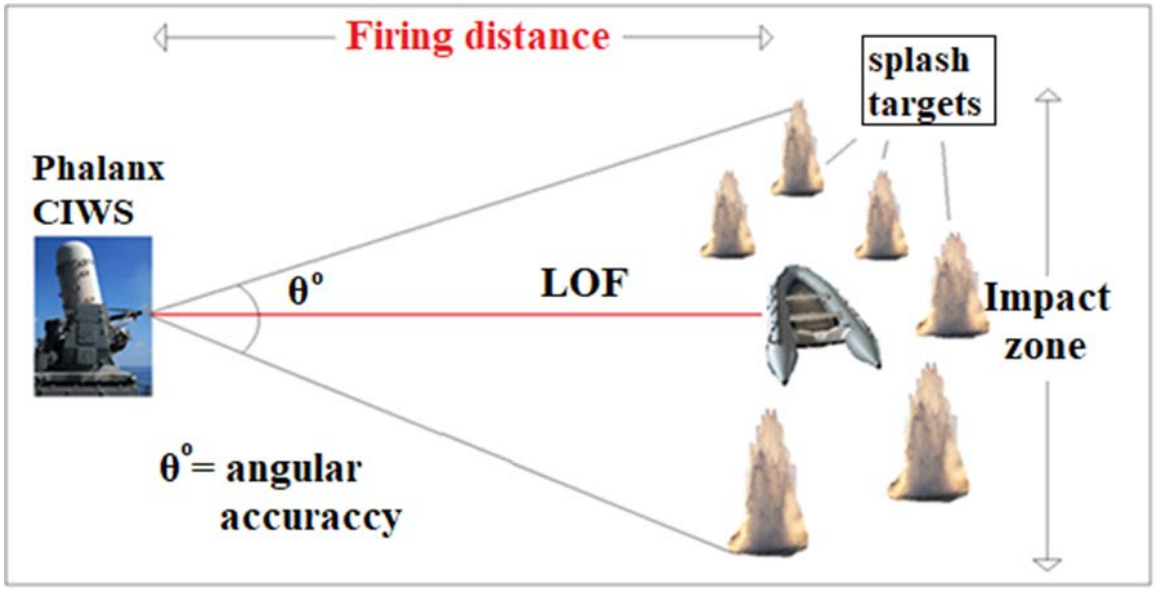

Figure 3. Firing Distance and Impact Zone. Adapted from [6].

To find R_{IZ}, we observed in Figure 1 that the bullets falling near the target produced a large splash of sea water, which we used to estimate the size of the impact zone. We assumed the size of the inflatable boat was five meters. With the impact zone estimated to be 20 times the size of the boat, we found that $R_{IZ} \approx 100$ m.

We next sought an estimate for the firing distance R_{FD}. By examining a successive sequence of video frames, we observed that all of the sea splashes appeared almost simultaneously within a time period of 0.1 s, which indicated that the firing burst also lasted

this same amount of time. We also estimated the flight time of the burst t_{flight} to be one second by noting the moment within the video when the sound of the burst began and when the rounds were observed in the impact zone. Knowing that the muzzle velocity u_{Bullet} for a typical CIWS, such as a Phalanx, is 1,200 m/s, we found that the firing distance was approximately

$$R_{FD} = u_{Bullet} \cdot t_{flight} = 1200 \text{ m}. \tag{2}$$

With approximations for R_{IZ} and R_{FD} in hand, we used Eq. (1) to find the angular accuracy for the CIWS to be $\theta_S = 4.76^{\circ}$. This result was found with the assumption that the target was stationary and must be adjusted for the case of a moving target. We do this by finding the angular displacement of the moving target with respect to the CIWS and subtracting this value from θ_S. To find the angular displacement of the moving target, we assume that it is moving perpendicular to the LOF at a speed of 40 knots, which is typical for an inflatable boat of this size. For the burst time period of 0.1 s, the boat is able to travel a distance R_{CD} of approximately two meters. The corresponding angular displacement ϕ of the moving target is then estimated from

$$\phi = \arctan\left(\frac{R_{CD}}{R_{FD}}\right) = 0.1^{\circ}. \tag{3}$$

The total angular accuracy θ_{total} of the system is the angular accuracy θ_S adjusted for the angular displacement of the moving target ϕ using

$$\theta_{total} = \theta_S - \phi. \tag{4}$$

For the Phalanx CIWS and using the estimated figures found by examining available video, we found the angular accuracy to be $\theta_{total} = 4.66^{\circ}$. Following the same analysis for the video available for other CIWS, we found the angular accuracy of the AK-630 and the GOALKEEPER were 4.75 degrees and 4.4 degrees, respectively. In Figures 4 and 5, we can see the bursts of these two CIWS systems. For the performance of our prototype, we adopted the more restrictive value of 4.4 degrees, or ± 2.2 degrees, for the angular accuracy specification.

Figure 4. AK-630 CIWS Operating against Boat. Adapted from [7].

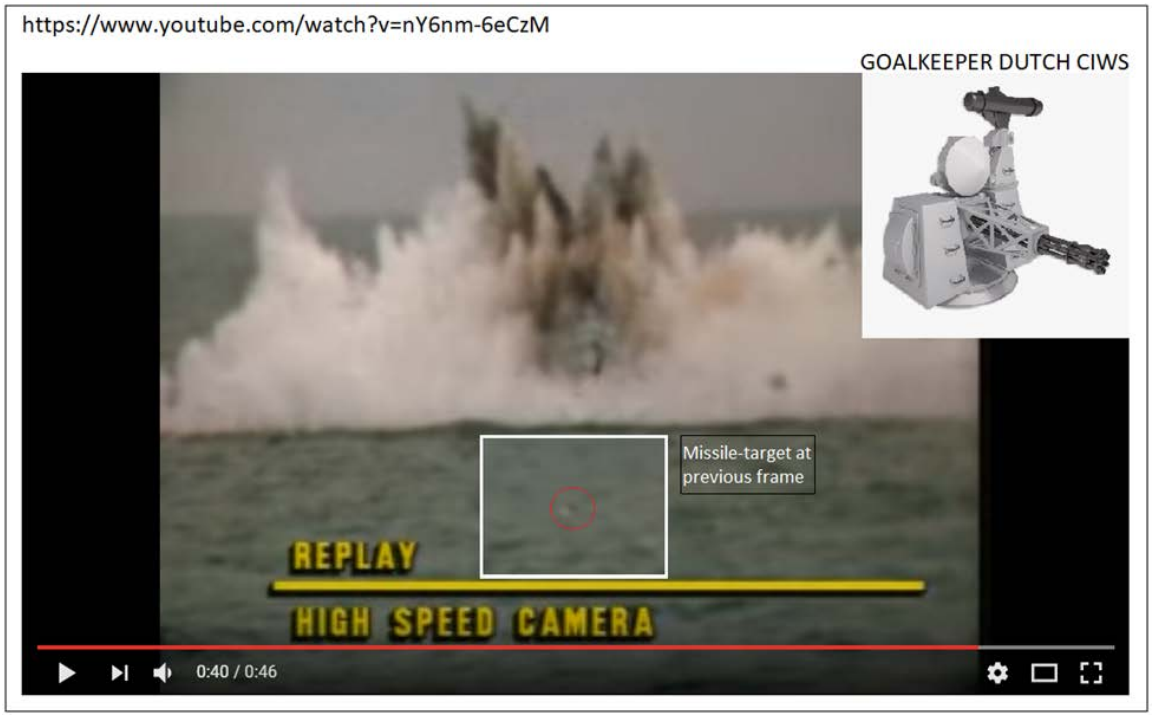

Figure 5. Goalkeeper CIWS against Missile Target. Adapted from [8].

B. SHIP MOTION MEASUREMENTS

There are three rotational motions that occur on ships, as shown in Figure 6. The tilting along the hull axis of the ship is called roll. The rotation perpendicular to ship's hull axis is called pitch and occurs fore and aft. Finally, yaw is the rotational motion around the ship's vertical axis. Depending on the sea state, the ship oscillates about each of these axes. The exact motion of a ship, however, is affected by many different factors and is too complicated to be described here. Nevertheless, to understand the dynamic environment in which our prototype CIWS is expected to operate, we carried out a series of measurements aboard ship. We used an Inertial Measurement Unit (IMU) to measure the motion of a 600-ton warship operating in normal sea conditions. The ship was an offshore patrol boat of the Hellenic navy in the Aegean Sea during the month of July 2017. The results of these measurements are used later in this thesis to estimate the system response of the prototype CIWS. In this section, we explain the experimental equipment used for this series of measurements and analyze the recorded data.

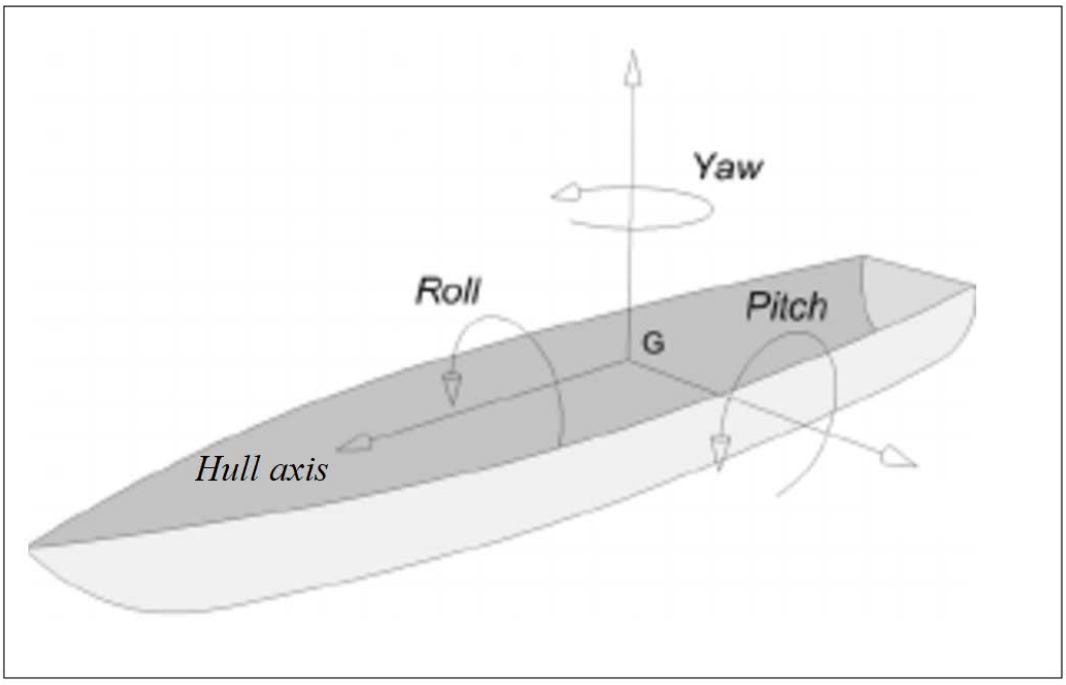

Figure 6. Ship Motion in Three Axes. Adapted from [9].

We captured the pitch and roll motion of the aforementioned patrol boat with an MPU-6050 IMU sensor and an Arduino microcontroller, as shown in Figure 7. The IMU-sensor measured the ship's natural motion and transferred the data to the Arduino serial monitor in real time using open source code. This code and the sensor's software library were available on Arduino's official website [10]. Then, we plotted the recorded angular motion using the MATLAB software. The ship's roll and pitch motion, in both the time and frequency domains, are shown in Figures 8 and 9, respectively. During the time period of the measurements shown in the figures, the ship operated in a sea state of 1.5 m. Additional measurements occurred during times of lower sea state conditions. Those results, however, are not included since the ship's motion was much less.

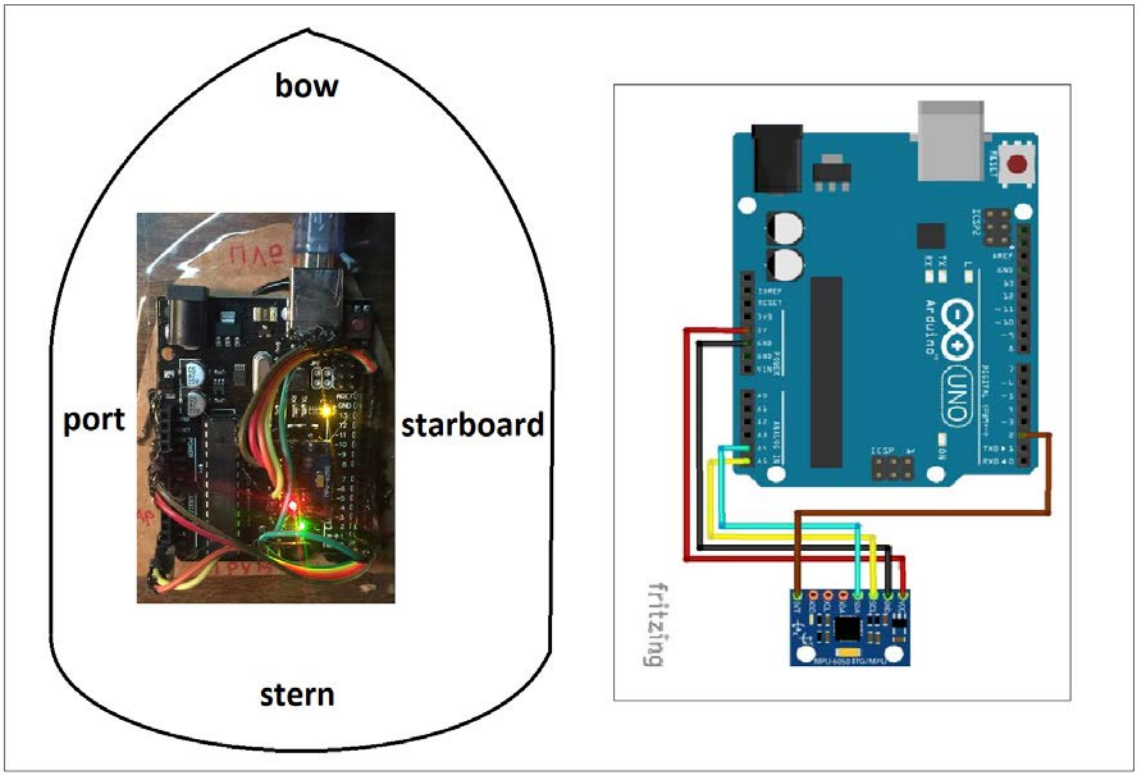

Figure 7. Ship's Motion Sensing Device. Adapted from [10].

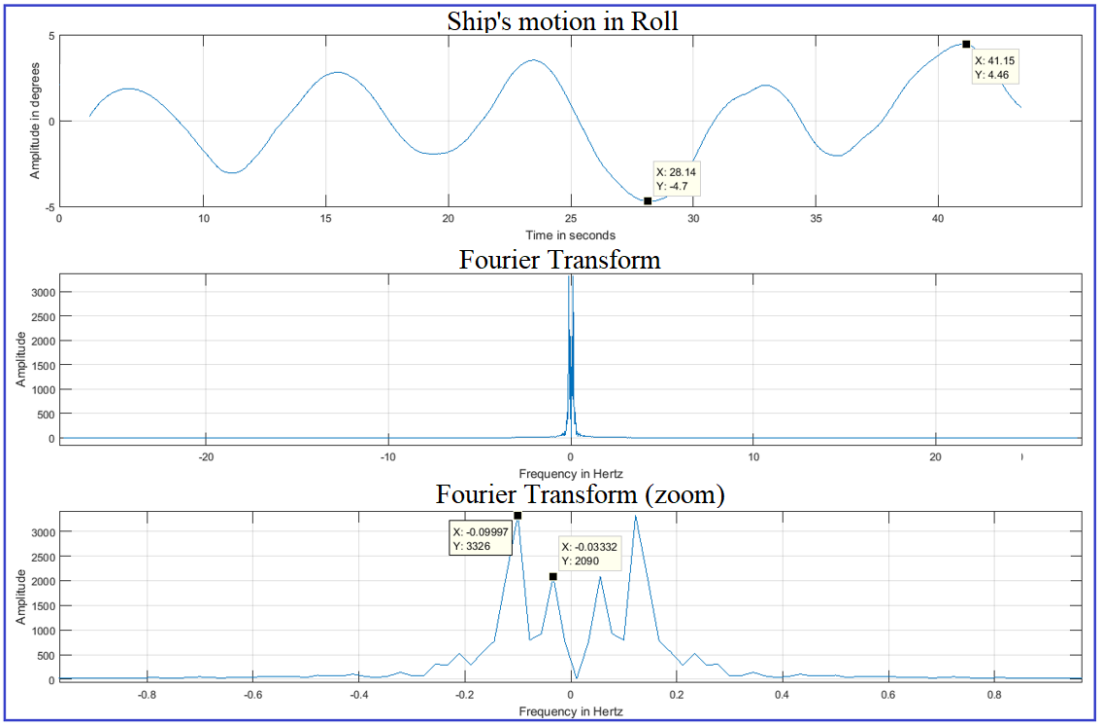

Figure 8. Ship's Roll Motion Analysis in Time and Frequency Domains

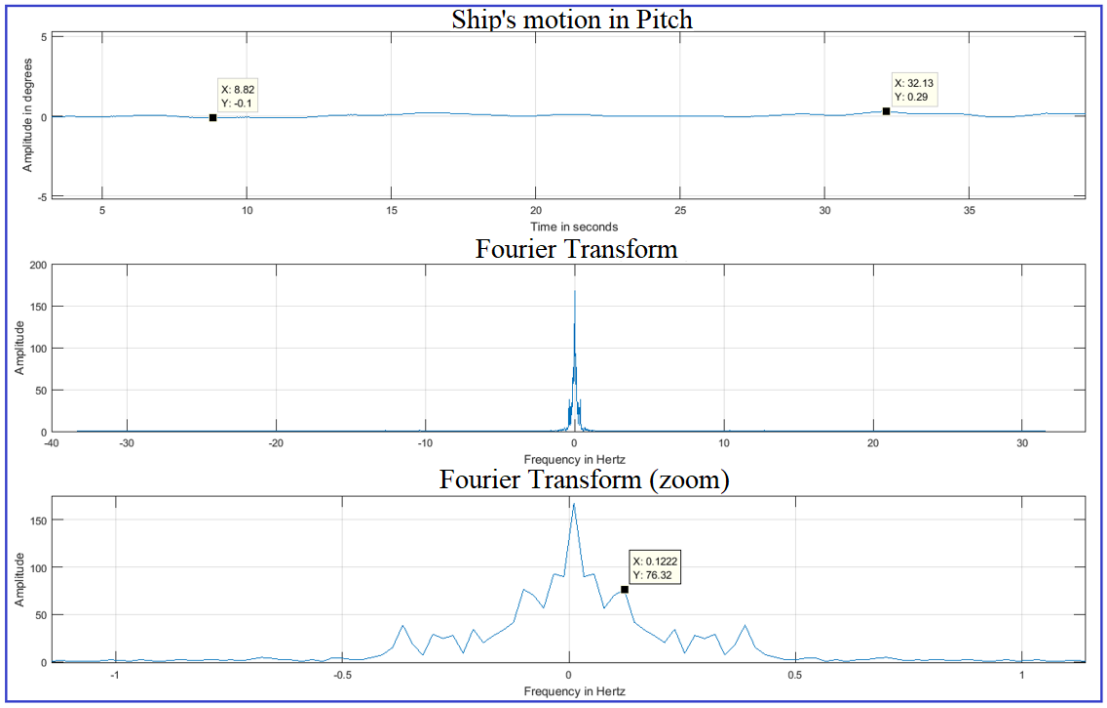

Figure 9. Ship's Pitch Motion Analysis in Time and Frequency Domains

As seen in the figures, the ship's motion is not exactly periodic; however, MATLAB's Discrete Fourier Transformation analysis tools enabled us to find the principal harmonics of the motion. The sampling period of the data is equal to the Arduino's loop execution time, which was measured to be 0.015 s. The roll motion had a dominant frequency of 0.1 Hz, which corresponds to a period of ten seconds, as shown in Figure 8. The amplitude of the ship's roll motion was almost 4.5 degrees. In the pitch axis, as shown in Figure 9, the measured frequency was 0.12 Hz. The measured amplitude of this motion, however, was less than 0.3 degrees. We use this frequency analysis later in this work to develop test signals for simulation and experimental testing.

C. DEGREES OF FREEDOM IN CIWS

Existing CIWSs have two Degrees-of-Freedom (DOF), and they do not use stabilizing platforms. Yet, they implement the pitch and roll angular corrections over the elevation and azimuth axis through integration with the ship's on-board sensors. The purpose of the elevation and azimuth axes is to reference the bullets' trajectory or LOF. In our prototype, we propose to implement four DOF. In this approach, a stabilizing platform will be designed to maintain a stable platform for the rest of the prototype's structure. This modification keeps the ship's gun safety sectors independent of the ship's motion and reduces the amount of integration required with the ship's on-board sensors.

All naval weapon systems have elevation and azimuth angle limits ensuring the safety of the ship and crew during firing operations. These limits prevent gun orientations beyond specified angles to avoid firing upon a ship's own structure. Consequently, this creates an unprotected sector above the ship, as shown in Figure 10. Additionally, the orientation of this sector changes according to the ship's motion, as shown in Figures 11 and 12, and creates a larger unprotected sector. By having a two-DOF independent stabilizer at the base of the system, we maintain the desired elevation and azimuth safety limits, and a CIWS with four DOF produces an unprotected sector that is independent of the ship's inclinations.

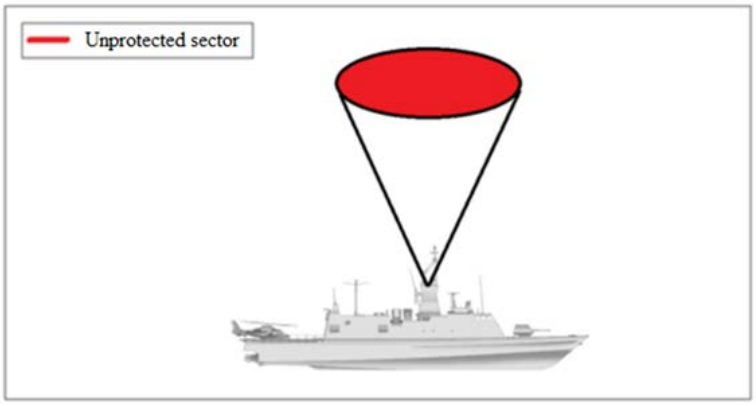

Figure 10. Unprotected Sector above a Battleship

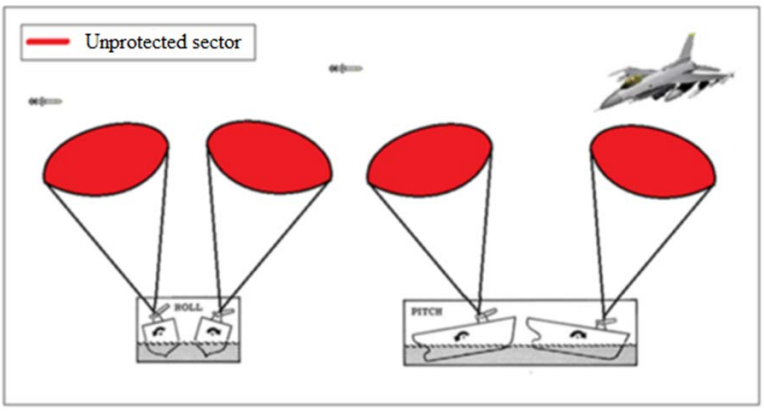

Figure 11. The Influence of the Ship's Motion over the Unprotected Sector

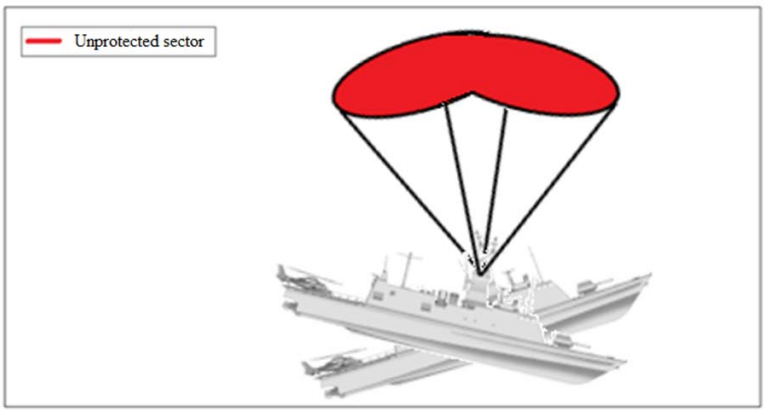

Figure 12. Aggregate of the Unprotected Sector Caused by the Ship's Motion

D. SUMMARY

First, from the analysis of available video of the operation of existing CIWS systems, we establish that the angular accuracy of our prototype CIWS should be 4.4 degrees, or ± 2.2 degrees. We also specify the dynamic operating environment to be 0.1 Hz, as was found through the experimental measurements aboard the patrol boat operating in the Aegean Sea. Lastly, we decide on a CIWS having four DOF to minimize the unprotected zone and reduce the integration required with a ship's on-board sensors.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN

In this chapter, we provide a description of the prototype CIWS design, hardware specifications, and controller design and implementation through software.

A. BRIEF DESIGN DESCRIPTION

The prototype CIWS consists of three major parts: a stabilizer at the base, a gun control unit at the top of the structure, and a computer unit, as shown in Figure 13. In this section, starting from the base and moving to the top of the structure, we explain the design characteristics of the prototype's components. The purpose of the stabilizer is to reduce the influence of the ship's motion during targeting and firing. The gun control unit, which contains the machine gun, is also equipped with a video camera to provide real-time video for the automatic control of the weapon's orientation over the angles of elevation and azimuth. Finally, a computer control unit receives the real-time video and performs video image processing to complete the automatic control of the CIWS. At this point, it is useful to describe the design of each part and to introduce the specifications of the prototype configuration.

Figure 13. Prototype's Overall Design in SolidWorks Software

To begin with, the stabilizer is an independent, two-DOF unit that compensates for the ship's pitch and roll motion, as shown in Figure 14. An MPU-6050 IMU sensor mounted at the top of the structure monitors the ship's motion and transfers the data to an Arduino microcontroller. Two servomotors serve to stabilize the two-level structure in the pitch and roll axes. The servomotors are controlled through a proportional-integral-derivative (PID) controller. The stabilizer is designed to operate independently from the other units of the prototype.

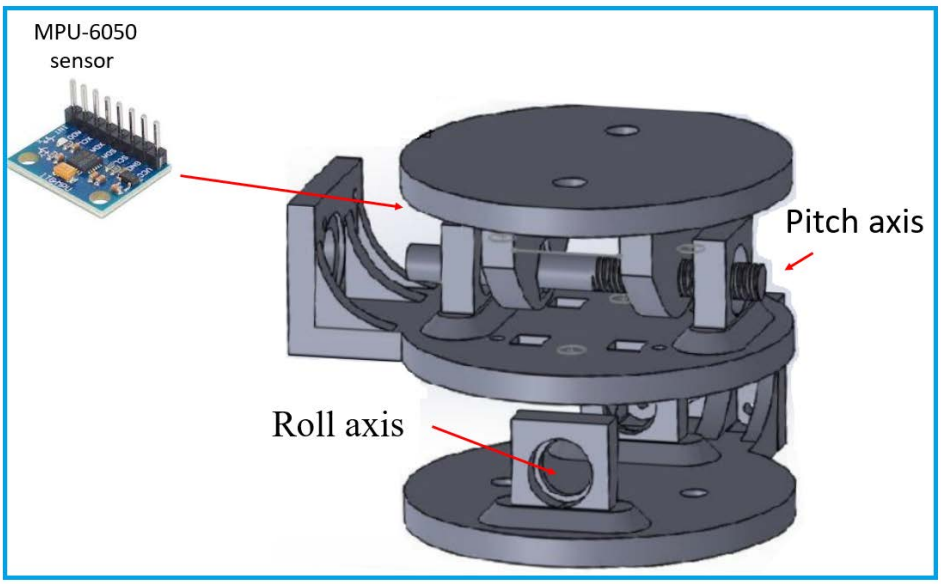

Figure 14. Stabilizer Design in SolidWorks Software

Secondly, the gun control unit is installed above the stabilizer; it operates totally independent of the unit below it. Similar to the stabilizer unit, it also has two-DOF, with two servomotors independently controlled over elevation and azimuth, as shown in Figure 15. Although not shown in the figure, a machine gun and a camera are attached to the gun control unit. The camera is installed at the front of the unit and aligned with the gun's muzzle, providing real-time video of the target to the computer unit. During each iteration of the control loop and once the computer vision process is completed, the Arduino microcontroller transfers the computed angular corrections for elevation and azimuth to the servomotors, as shown in Figure 16.

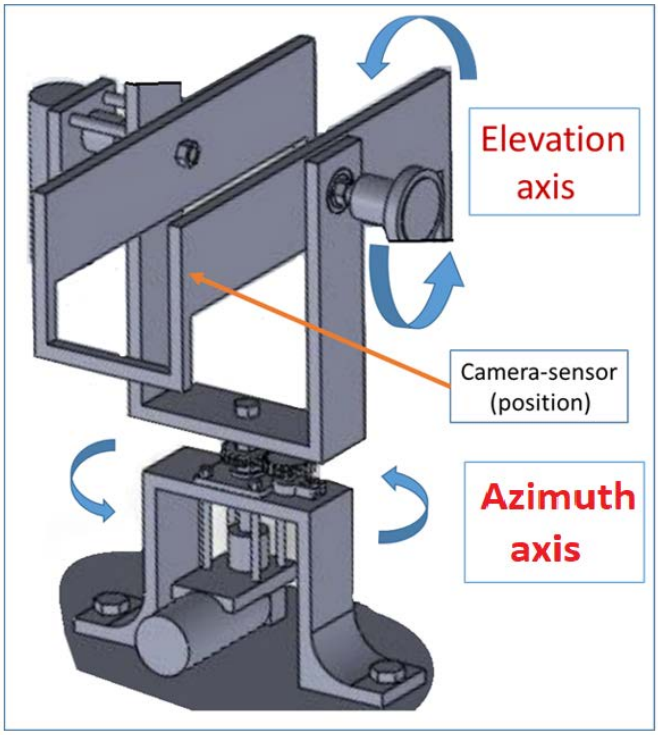

Figure 15. Gun Control Unit Design in SolidWorks Software

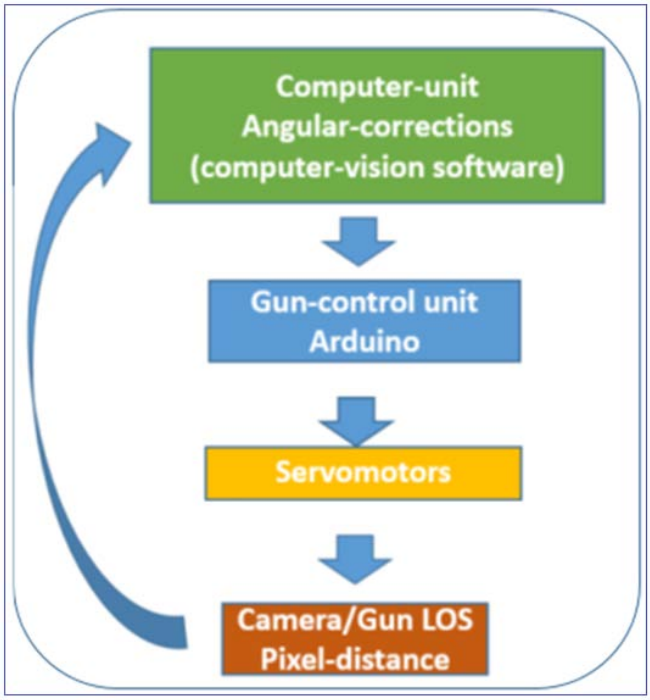

Figure 16. Gun Control and Computer Unit Closed-Loop Block Diagram

The third and final component of the prototype CIWS is the computer unit. Its main role is to determine the angular difference in elevation and azimuth between the system's LOS and the two-dimensional (2D) image position of the target using the video image frames it receives from the camera on the gun control unit. A typical image frame is shown in Figure 17, which also illustrates the image center highlighted with a red cross cursor and a designated target identified with a small green square. The image processing algorithm utilized within the computer unit computes the pixel difference between the red cross cursor and the green target designator. This pixel difference is then used to correct the elevation and azimuth angles of the gun control unit to maintain the LOS aligned with the LOF. The operator provides an initial indication of the target to the system by selecting a specific pixel area in the image frame.

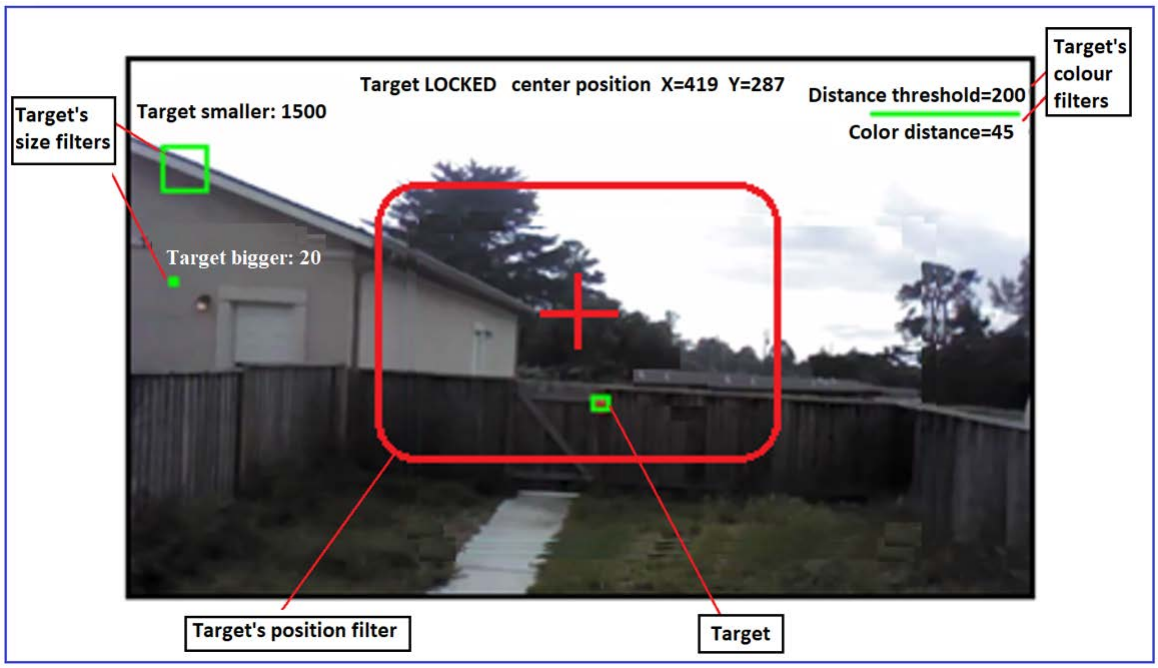

Figure 17. Computer Unit Video Frame

B. HARDWARE

In this section, we describe in more detail the hardware components used in the construction of the prototype CIWS.

1. Body and Material

The initial design of the CIWS prototype used a metal support structure, as shown in Figure 18; however, this building material was soon abandoned because it was expensive and not easy to modify. The final prototype is a three-dimensional (3D)-printed structure made of polylactide (PLA). The material is biodegradable, is much easier to modify, and much less costly to produce. The CIWS prototype is shown in Figure 19. Both the stabilizer and the gun control unit are constructed with 3D-printed parts. Four DC servomotors, an MPU-6050 IMU sensor, and two Arduino microcontrollers are installed and integrated to give the assembly four DOF. The figure also shows a DC power supply located at the lower-left corner. A Hewlett-Packard ENVY m7 Notebook laptop computer, not shown in the figure, serves as the system's computer unit.

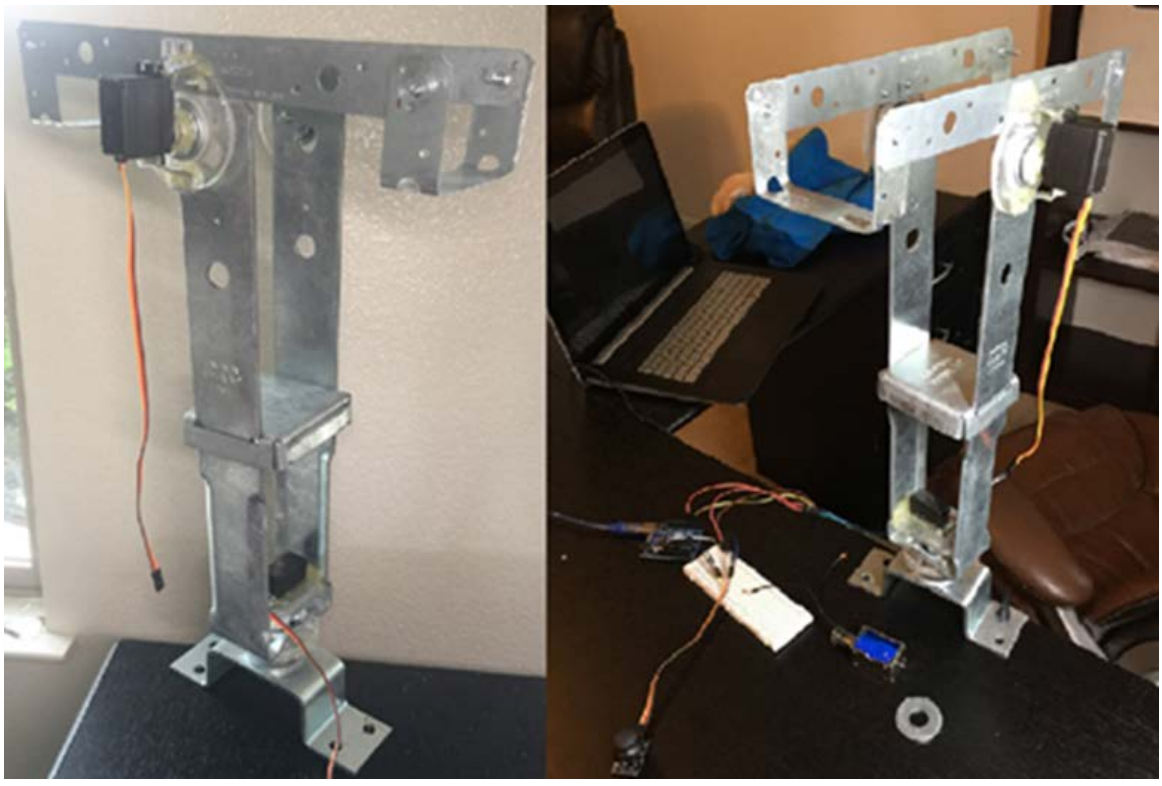

Figure 18. Initial Model

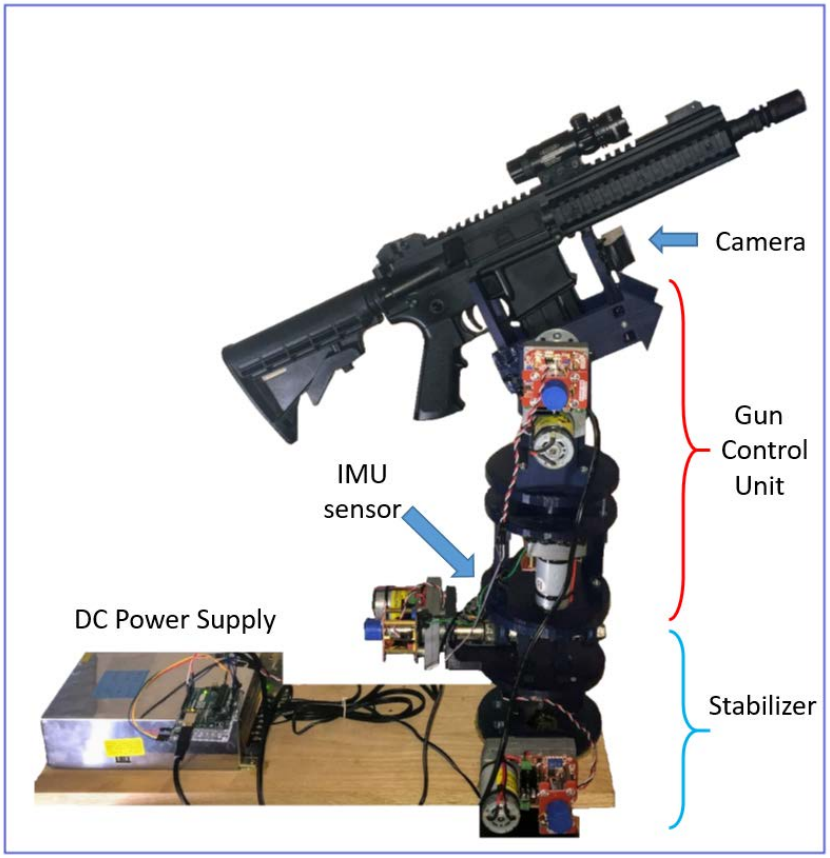

Figure 19. Final Prototype Model

2. Stabilizer

The maritime versions of any weapon system must provide a stabilization mechanism to compensate for the ship's motion. A ship's gyroscope senses the natural motion of the ship and provides its orientation. Usually, naval weapon systems have two DOF to control their orientation in elevation and azimuth. Additionally, to compensate for the ship's motion, pitch and roll corrections are also applied to these two axes. In our four-DOF prototype, we implement an automatic stabilizer to achieve the desired stabilization, as shown in Figure 20. We chose this architecture because it addresses the need for portability in our prototype design. This base consists of an IMU sensor, two servomotors, and an Arduino microcontroller. In the next paragraphs, we explain the purpose of the four-DOF design and describe the electric/electronic components used in the assembly of the base.

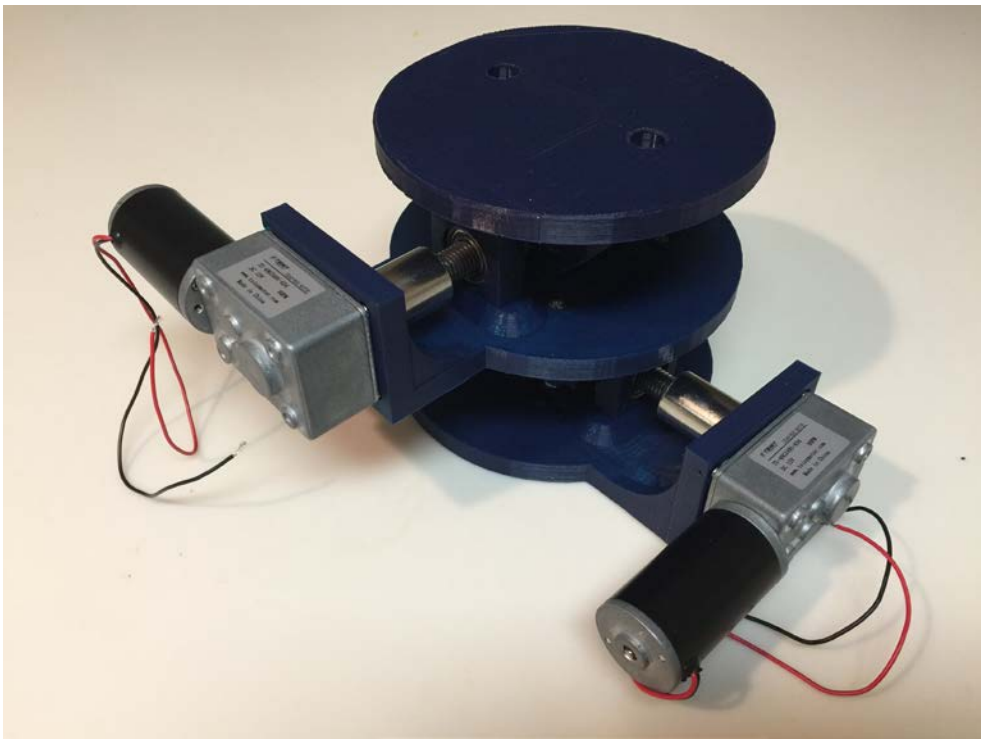

Figure 20. Stabilizer Base Hardware

In our design, the function of the ship's gyroscope is achieved with the MPU-6050 IMU providing the desired portability. IMUs are electronic devices that consist of gyroscopes and accelerometers and are used to measure the movement of a body [11]. The MPU-6050 IMU is a six-DOF motion tracking device that combines a three-axis gyroscope and a three-axis accelerometer. The accelerometer measures the inertial forces that occur during the movement of the device. Naturally, it is extremely sensitive to vibrations and mechanical noise. The gyroscope provides the angular velocities of the device with respect to each of the sensing axes. While they are less sensitive to mechanical noise than the accelerometers, the gyroscopic drift limits their performance. Finally, the combination of those two sensor types decreases the total orientation error [1]. For the purpose of our research, a single MPU-6050 IMU is sufficient to demonstrate our concept despite the aforementioned errors. This sensor is attached to the upper plate of the stabilizer and communicates with a connected Arduino microcontroller, as shown in Figure 21, to provide the orientation measurements for the control loop.

The Arduino is a single-board microcontroller that combines software and hardware, which simplifies the creation of interactive electronic projects [2]. This open-source platform is controlled through user developed software using C and C++ programming languages. Digital and analog input-output channels on the Arduino board allow communications with different sensors and other electronic devices. The Arduino microcontroller is at the heart of the prototype CIWS control loop. It receives the IMU's measurements and translates them into the angular corrections which command the motion of the attached servomotors to align the platform with the sea surface.

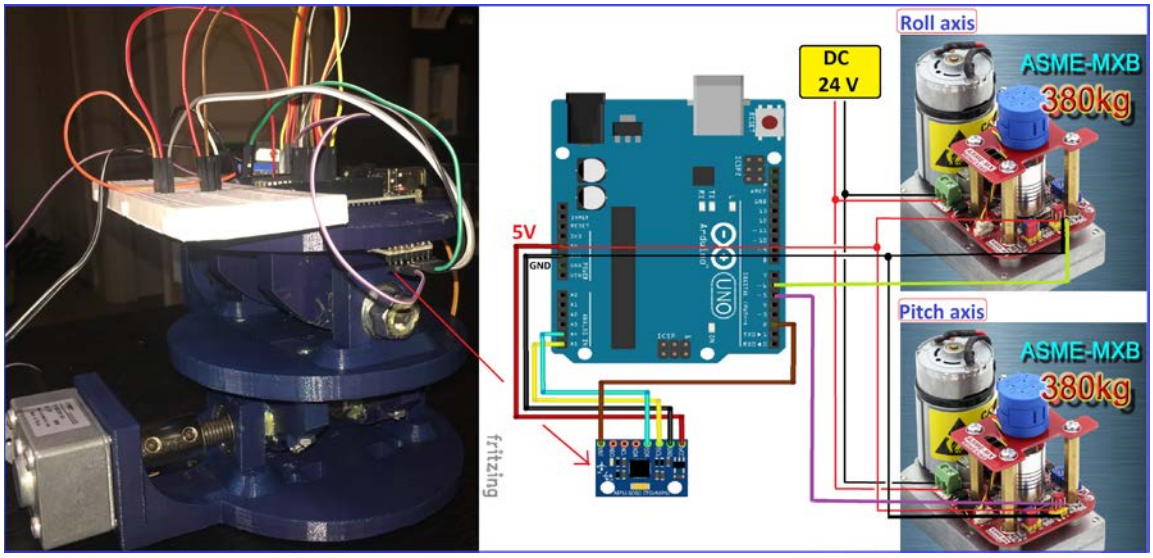

Figure 21. Stabilizer Configuration and Schematics. Adapted from [12], [13].

Subsequently, the stabilizer's servomotors received the angular corrections through the transferred pulse-width modulation (PWM) signal. Servomotors are a combination of a rotary encoder, a sophisticated controller, and a DC motor. Applying a discrete signal to a servomotor, we accurately control its shaft's angular position. This accuracy is achieved through a closed-loop process built into the ASME-MXB servomotor assembly. Specifications of the ASME-MXB servomotors that we use for the prototype are shown in Figure 22. The output shaft of each servomotor is attached to each of the pitch and roll axes, respectively, to control the orientation of the base's upper mounting plate.

ASME-MXB

RECOMMENDED OPERATING CONDITIONS

Parameter		
Operating Voltage	12V~24V/DC (11V/MIN~30V/MAX)	
No-load current	<500mA	
Limit Angle	±1800° or 0°~3600° (MAX)	
Control mode	R C mode	1ms pulse mode:1ms~2ms(pulse width)
		2ms pulse mode:0.5ms~2.5ms(pulse width)
	voltage mode	0V-5V input voltage (0.22V to 0°,4.78V to 3600°)
Material	Gear and shaft: alloy steel; gear box: aluminum	
Weight	600g	
Dimension	95mm*60mm*110mm	
Mechanical Life	> 1,000,000/full load (conservative values)	

ABSOLUTE MAXIMUM RATINGS

Parameter		
limit input voltage	11V~30V/DC (MAX)	
limited output current	5A±0.2A (MAX)	
limit Angle	±1800° or 0°~3600° (MAX)	
Control mode	R C mode	1ms pulse mode : 1ms~2ms(pulse width)
		2ms pulse mode : 0.5ms~2.5ms(pulse width)
	voltage mode	0V-5V input voltage (0.22V to 0°,4.78V to 3600°)
Weight	550g~600g	

ASME-MXB	380kg.cm max	340kg.cm max	0.5s/60°MAX

Figure 22. ASME-MXB Servo Characteristics. Adapted from [14].

3. Gun Control Unit and Computer Unit

The gun control unit and the computer unit work together to maintain the orientation of the gun. As with the stabilizer unit, the gun control unit also utilizes two ASME-MXB servomotors, as well as an Arduino microcontroller, and a web camera, as

shown in Figure 23. First, the attached laptop computer or computer unit provides the image processing resources needed for this component. Secondly, the camera is aligned with the gun's barrel and transfers real-time video to the laptop computer. Thirdly, the laptop computer unit estimates the elevation and azimuth alignment errors of the gun control unit through an image processing algorithm. Next, it translates these errors into angular corrections for the elevation and azimuth axes and transfers the results to the Arduino microcontroller. Finally, the Arduino software controls the two servomotors to align the gun barrel with the target's position. This is a closed-loop, self-correcting process that occurs once in every processing loop to keep the gun aligned to the target while it is moving.

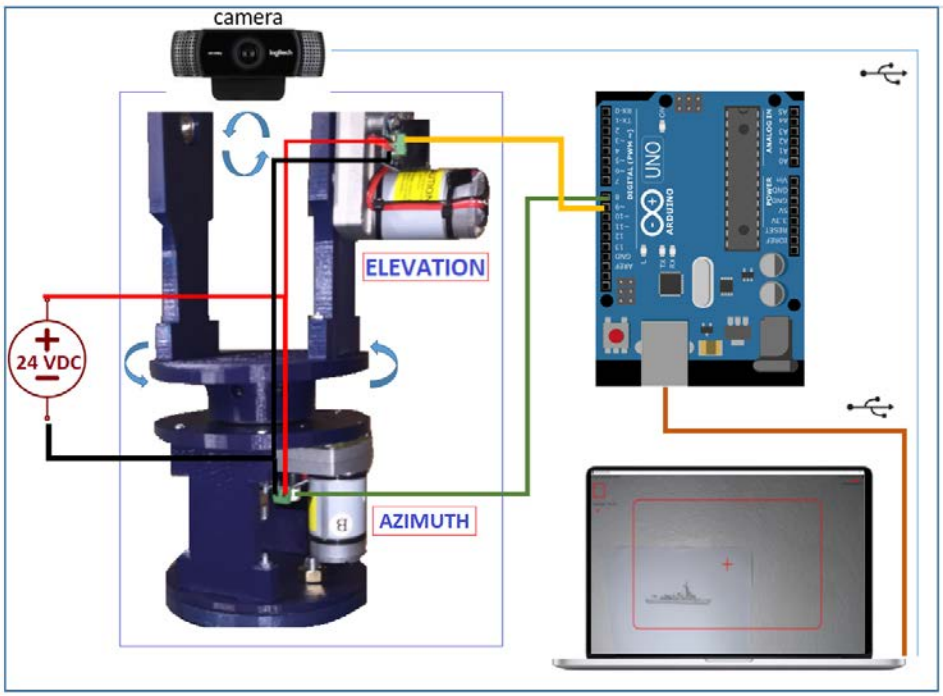

Figure 23. Gun Control Unit Hardware and Schematics

C. THEORY AND IMPLEMENTATION OF CLOSED-LOOP SYSTEMS, PID CONTROLLERS, AND COMPUTER VISION

Now that we have introduced the hardware components of the prototype, in this section, we explain how the control theory is implemented in the prototype. The prototype uses four servomotors, one for each DOF. While the stabilizer must counterbalance the ship's motion in the pitch and roll directions, the gun control unit must align the barrel of the gun with the target over the elevation and azimuth angles. The overall tracking accuracy of the prototype is determined by the accuracy with which the motors can be controlled. To improve the CIWS response, we created a closed-loop PID controller for each of the motors. In this section, we review the fundamentals of closed-loop systems and PID controllers, and we explain how they are applied to the prototype.

1. Closed-Loop Systems

In engineering applications, we design feedback controllers to achieve higher accuracy in a system [15]. Disturbances can negatively influence and alter the output of a system. In a closed-loop system, the output is combined with the input to eliminate the influence of the disturbances [15]. For our prototype, we created a closed-loop control system for each of the four DOF: elevation, azimuth, pitch, and roll. With one motor controlling the motion of each DOF, these closed loops create a self-correcting system to achieve higher accuracy while tracking a target. The application of closed-loop control theory to the prototype's main parts is shown in Figure 24. Within the next paragraphs, we describe the closed-loop control theory implementation for the two-DOF stabilizer and the two-DOF gun control unit.

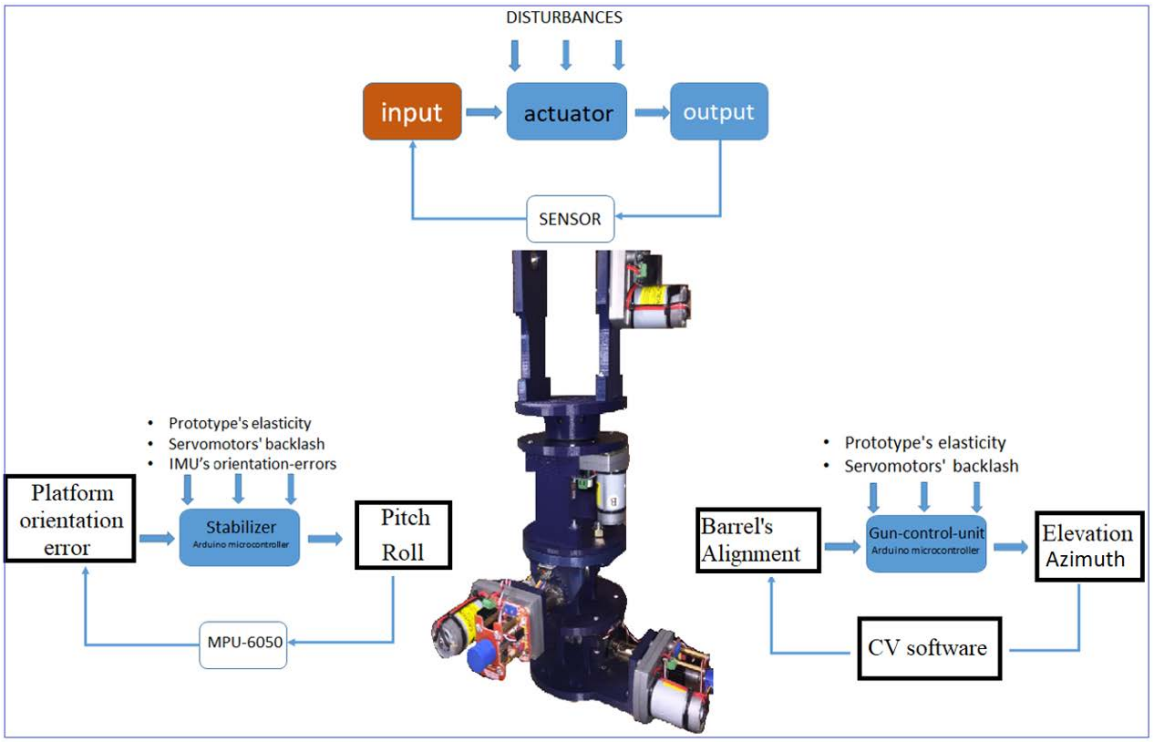

Figure 24. Closed-Loop Control Theory Implementation within the Prototype

During the initial design of the stabilizer, we installed the IMU sensor at the bottom of the structure, monitoring the ship's pitch and roll orientation angles; however, the observed response was insufficient and resulted in an average error θ_{error} of four degrees. While the purpose of the stabilizer is to ensure that the ship's natural motion does not influence the gun, this angular error resulted in a missed target error D_{error} of as much as 84 m at an assumed firing distance R_{FD} of 1200 m, estimated by

$$D_{error} = R_{FD} \cdot \tan(\theta_{error}) \ . \tag{5}$$

The control system approach described in the preceding paragraph essentially represents an open-loop control system due to the lack of a feedback loop. The sensor monitored the inclinations of the lower base and the controller adjusted the inclination of the upper base; however, the upper base inclination was not measured and compared to the desired orientation, resulting in the poor response. The need for higher accuracy led to the development of a feedback control loop that reduces the negative influence of the disturbances. By relocating the IMU sensor to the upper plate of the structure, we created

a closed-loop control that improves the stabilizing accuracy. Through this modification, the IMU sensor monitors the orientation of the upper base in each processing loop and results in a reduced total angular error. The block diagrams of the open and closed-loop control configurations are shown in Figure 25.

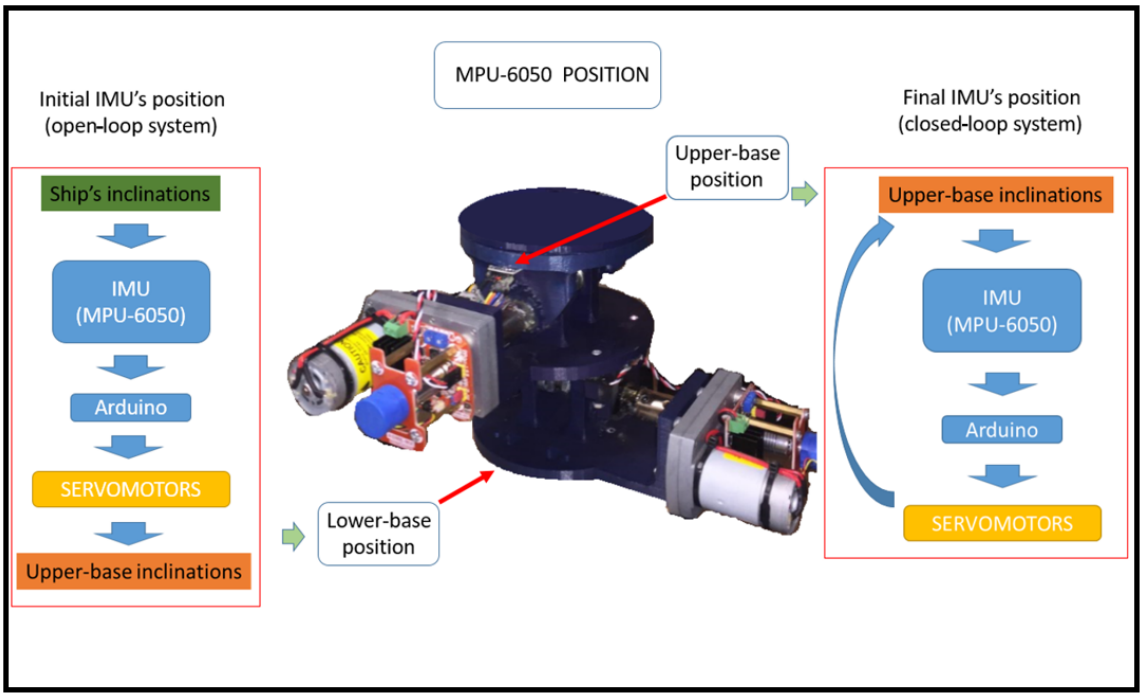

Figure 25. Open- and Closed-Loop Stabilizer Configurations

The closed-loop design is also utilized in the gun control unit. The main purpose of this unit is to align the gun's barrel with the target. The gun and a camera are installed at the top of the structure aligned to each other and facing the target, as previously shown in Figure 19. This configuration ensures that the gun and the camera always point to the same spot or target in the LOF direction. The camera transfers real-time video of the target to the computer unit. The video images are processed to determine the pixel distance between the target and the center of the image, as shown in Figure 26. Subsequently, the pixel distance is converted into the appropriate angular corrections for elevation and azimuth. In turn, these corrections are transferred to the Arduino located on the gun control unit, which

controls the azimuth and elevation servomotors to maintain the LOF aligned with the target. This closed-loop process was previously shown in Figure 16.

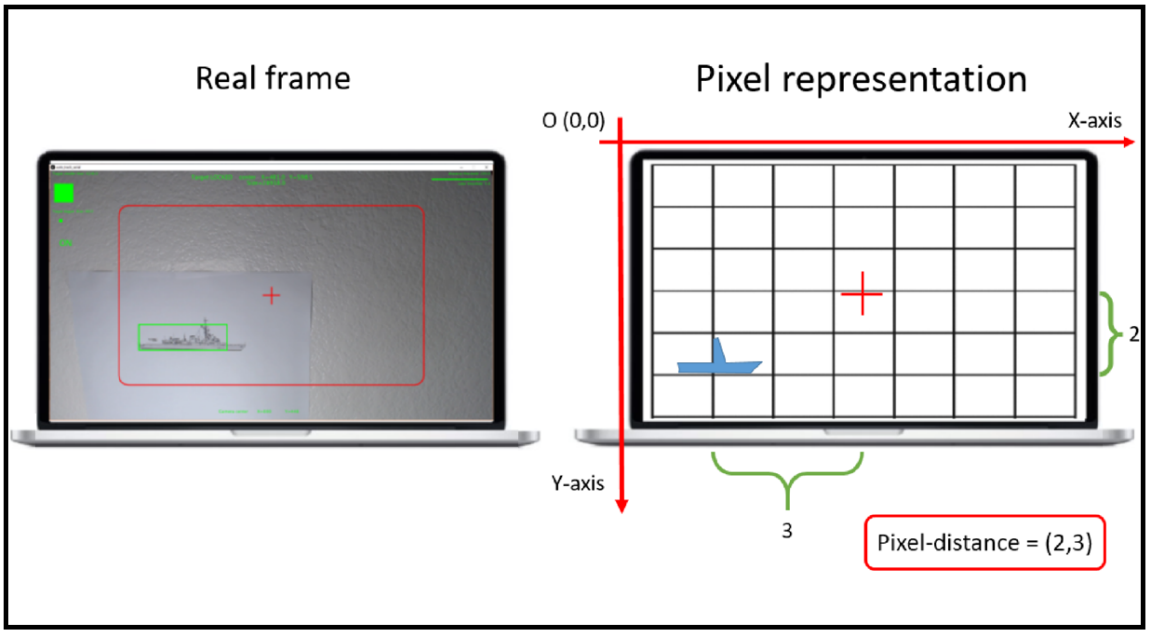

Figure 26. Pixel Distance between Target and Image's Center

In each processing loop, the computer unit determines the current pixel distance found in the last video frame. The feedback in the aforementioned procedure is the target-center pixel distance. A zero pixel distance means that the barrel and the camera are perfectly aligned with the LOF directed at the target. At this point, it is crucial to mention that since the gun control unit is installed at the top of the stabilizing platform, the angular stabilizing errors also influence the pixel distance. Eventually, the factors that influence the accuracy of the gun control unit are the angular errors in the stabilizer unit and the target's relative motion. Nonetheless, this closed-loop system minimizes these errors and drives the tracking error to zero. The prototype's block diagram of the overall closed-loop system is shown in Figure 27.

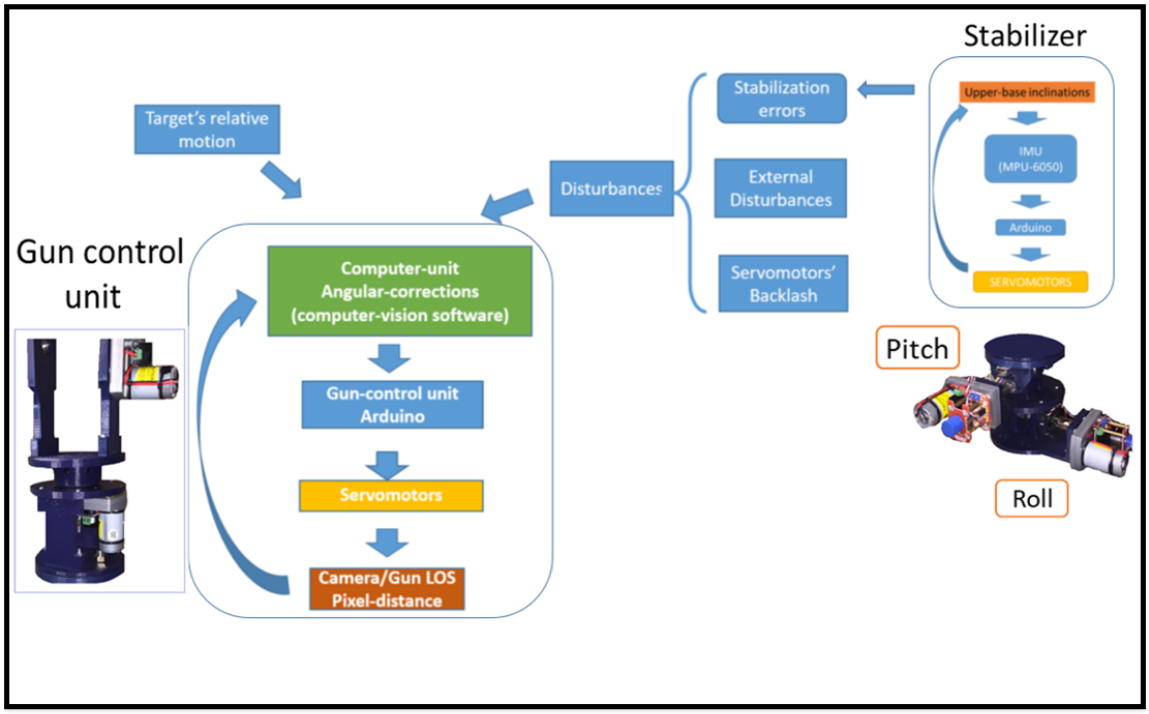

Figure 27. Prototype's Overall Closed-Loop Block Diagram

2. PID Control from Theory to Implementation

PID controllers are feedback-control loops that provide continuous and accurate control of an engineering system or process [15]. A real-life example of such a controller is the cruise control that maintains the desired speed of a car by adjusting the engine's fuel flow. In our prototype, we implemented PID control loops for each of the motors to achieve higher control accuracy in each DOF. Through this feedback control process, the servomotor's shaft angular position is adjusted automatically to maintain the desired pitch, roll, elevation, or azimuth angle. In this section, we review PID control theory, and we explain how it is implemented in the prototype.

Specifically, a PID controller is a closed-loop algorithm in which the input is the difference between the desired and the current state of a plant or process [15]. The controller aims to eliminate this difference, which is called the error $e(t)$. The user and/or other sensors in the system determine the desired state of the mechanism, which is called

the set point. The main role of the controller is to reduce the difference between the present and the desired angular position of the plant. The block diagram of a PID controller that regulates the angular position of a process is shown in Figure 28.

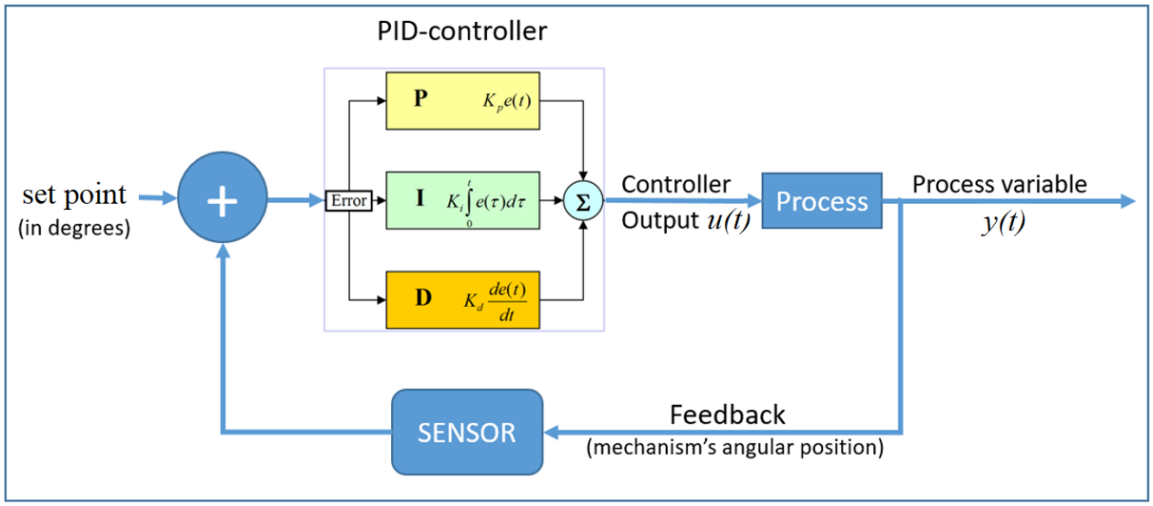

Figure 28. PID Controller Block Diagram

The PID controller is made up of three terms: the proportional term, the integral term, and the derivative term. The mathematical expression for the PID controller is

$$u(t) = P + I + D \ , \tag{6}$$

where the proportional term P is given by

$$P = K_p \cdot e(t) \ , \tag{7}$$

the integral term I is

$$I = K_i \cdot \int_0^t e(\tau)d\tau \ , \tag{8}$$

and the derivative term D is

$$D = K_d \cdot \frac{de(t)}{dt} \ . \tag{9}$$

Each of the terms in Eq. (6) influences the controller's output $u(t)$. The coefficients K_p, K_i, and K_d are the adjustable weights or gains for each of the terms in Eq. (6) and are tuned to achieve a desired overall system response.

Two PID controllers are used in the stabilizer to regulate the orientation of the upper plate. The first PID controller maintains the pitch of the plate by adjusting the angular position of the output shaft of the lower motor. A second PID controller maintains the roll angle of the stabilizer upper plate. The IMU sensor is installed on the upper plate and computes the orientation of the plate so that the PID controllers can maintain a zero pitch and roll orientation. The PID controllers are implemented in software on the Arduino microcontroller. Consequently, the controllers are discrete algorithms with a measured loop period of 0.015 s.

3. Computer Vision

Computer Vision (CV) integrates computer software and hardware [16] to extract information from images. In military applications, CV has a major role since cameras are passive sensors, and an enemy cannot easily detect them. Almost all modern weapon systems have CV capabilities that serve to enhance their ability to detect a target. In our prototype, a CV software program estimates the alignment error between the LOS of the gun control unit and the LOF.

Using an image processing algorithm available from open-sourced CV software [17], we processed individual digital image frames that were captured by the gun control unit's camera. The pixels in each frame had a value ranging from zero to 255 that represented their image intensity from black to white, respectively. Using these values the CV algorithm executes the following steps. First, the computer unit's operator designates a target by selecting a 3-by-3 pixel area on the image [17]. Second, the algorithm identifies the target by tracking the average color value within this area in the image frame. Third, the algorithm computes the pixel distance P_{dist} between the center of this target and the center of the image. Then, P_{dist} is translated to the angular correction θ_{cor} given the camera's field-of-view ϕ_{view} and the image's total pixel width W_{pixel}. In Figure 29, the relation between the pixel distance and the angular correction can be seen. Mathematically, this is described by

$$\theta_{cor} = \arctan\left[\frac{2P_{dist}}{W_{pixel}}\tan\left(\frac{\phi_{view}}{2}\right)\right]. \tag{10}$$

Once θ_{cor} is computed in each processing loop, the value is sent to the Arduino microcontroller where the PID controller computes the appropriate motion for the servomotors to regulate the elevation and azimuth angles.

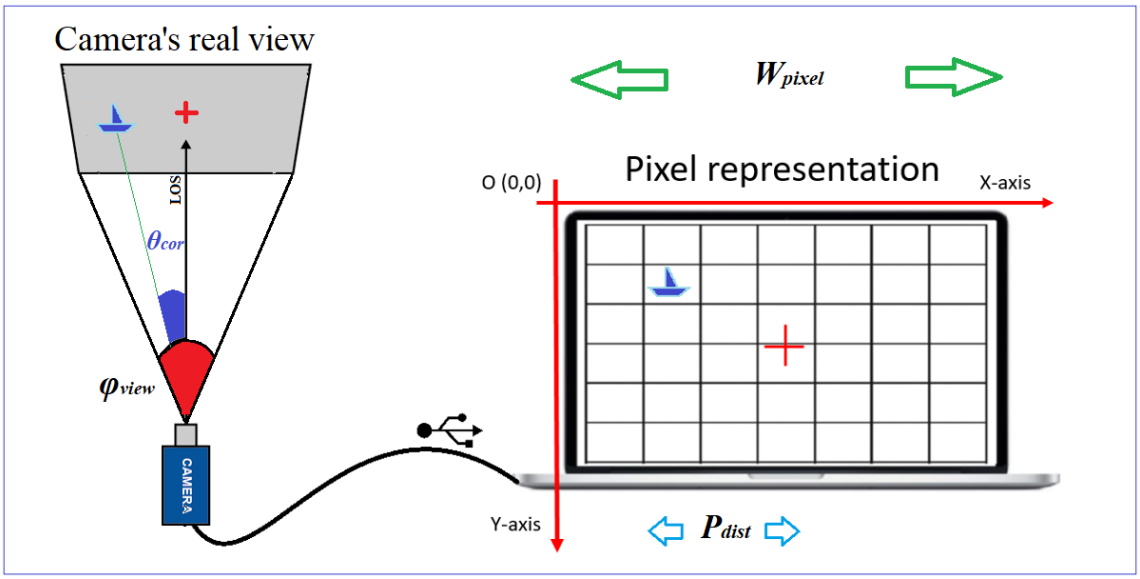

Figure 29. Pixel Distance Translated to Angular Correction

D. SOFTWARE

In this section we describe the three primary software programs developed for this thesis project and discuss how they interface with the hardware. Two of the programs were developed for the Arduino microcontrollers, which are installed in the stabilizer unit and the gun control unit. The third software program is operated on the computer unit (laptop) executing the image processing algorithm. In our prototype, the stabilizer software is an Arduino program [18], also known as a *sketch*, which implements two PID controllers that control the pitch and roll servomotors. The gun control unit's software is also an Arduino sketch that establishes the communication between this unit and the computer unit. The computer unit's software is a CV program developed in the Processing programming

language that visualizes the data and interacts with the user [17]. The last two software programs communicate with each other enabling the precise control of the gun control unit.

1. Arduino Code for the Stabilizer

For the stabilizer software, we implement Arduino code to contain three major functions: 1) to read the IMU's orientation, 2) to compute the optimum servomotor performance using a PID algorithm, and 3) to transfer the angular corrections to the servomotors. These three basic functions occur during every processing loop to keep the IMU that was installed on the upper plate aligned with the sea surface. To read the sensor's data we include the *i2cdev* library. Furthermore, we integrate a significant part of the example code named *MPU6050_DMP6* from the Arduino official site [18]. A flow chart of the stabilizer's code is shown in Figure 30.

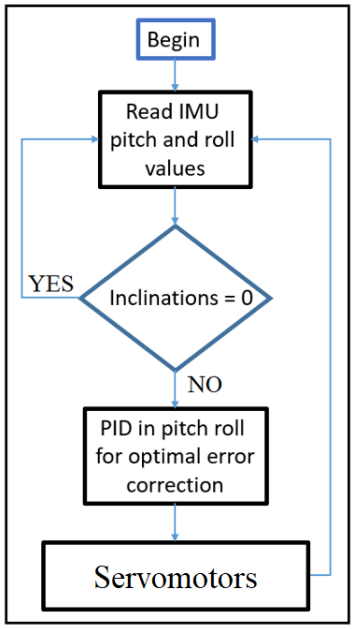

Figure 30. Stabilizer Code Flow Chart

2. Arduino Code for the Gun Control Unit

In the gun control unit, we require an Arduino sketch that communicate with the computer unit and operate the servomotors for the elevation and azimuth angles. The code

receives the elevation and azimuth corrections from the computer unit and transfers them to the servomotors. These data are encoded by the computer unit's software within a code word that has specific characters located in certain positions. To avoid any communication errors between the computer unit and the gun control unit, the received data is first decoded and then the order in which the characters were received is verified. A flow chart of the gun control unit's code is shown in Figure 31.

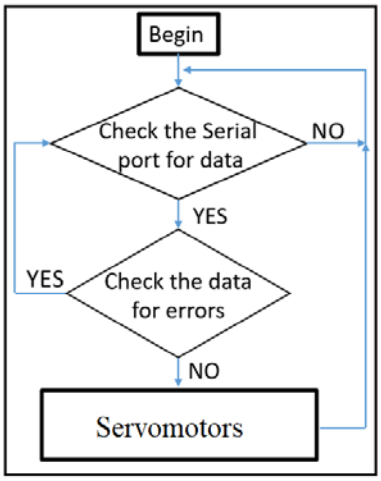

Figure 31. Gun Control Unit's Code Flow Chart

3. Code for the Computer Unit

The computer unit code is written in the Processing programming language and installed in the computer unit devoted to executing the CV algorithm. Parts of this code adjust several tracking and detection settings and determine the pixel distance between the target and the center of the image. Other functions facilitate the operator's real-time interface with the system. Finally, two PID controllers estimate the optimal angular corrections in elevation and azimuth. These corrections are transferred to the Arduino microcontroller's software. Several parts of this code were partially adapted from Daniel Shiffman's online tutorials sequence [17]. A simplified flow chart of the code is shown in Figure 32.

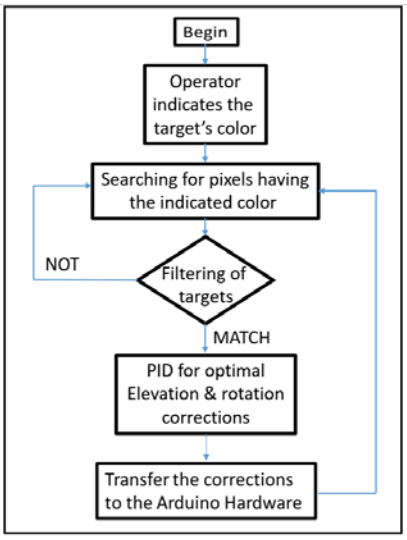

Figure 32. Computer Unit's Code Flow Chart

E. SUMMARY

A closed-loop system is a system that includes feedback control. A simple closed-loop system is composed of a sensor, a controller, and an engineering process or plant mechanism. To accurately control a closed-loop system, we can implement a PID controller in software within the controller processing unit. For this prototype, we implemented four PID controllers to regulate each of the DOF. The pitch and roll controllers were implemented in the Arduino software of the stabilizer. An IMU sensor that was installed on the upper plate of the unit provided the sea surface reference. Then the two controllers worked to maintain the orientation of the upper plate aligned with this reference as the base was free to pitch and roll. The gun control unit was installed on this upper stabilizer plate. In a similar way, the elevation and azimuth PID controllers trimmed the unit's LOS. A camera sensor installed on the gun control unit provided digital frames to the computer unit. Through the computer unit's software we estimated the pixel distance between the target and the image's center. This distance is the feedback signal that is provided to the Arduino microcontroller of the gun control unit.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. EXPERIMENTAL RESULTS AND SIMULATION

In this chapter, we evaluate the performance of the prototype through experimental measurement as well as simulation. The chapter is divided into two parts. In the first part, experimental measurements provide the response of the CIWS as it was stimulated with ship motion inputs similar to those described in Chapter II. Through this study, we examine the performance of the portable CIWS for ships of a size similar to the aforementioned patrol boat. In the second part, we use a simulation model to evaluate the potential performance of the prototype under a wider range of ship motions as these inputs were difficult to reproduce in the laboratory without the availability of specialized test equipment.

A. EXPERIMENTAL MEASUREMENTS

Experimental measurements were taken to evaluate the tracking accuracy of the prototype system and for comparison with the performance of existing CIWSs operating on smaller ship platforms. In this section, we initially evaluate the stabilizer's performance during simulated ship motion as described in Chapter II, Section B. Then we evaluate the gun control unit's performance as it tracks a moving target. The accuracy results in each set of measurements are compared to the accuracy of existing CIWSs. Through this comparison, we determine whether it is feasible to create a portable CIWS for smaller ship platforms.

1. Stabilizer Performance

To evaluate the performance of the stabilizer, the prototype was installed on a wooden base, as shown in Figure 19. We manually moved the stabilizer base back and forth to simulate the natural ship motion presented in Chapter II, Section B. As we described in that chapter, the roll component of the ship's motion was very similar to that of a sinusoid with a frequency of 0.1 Hz and a maximum amplitude of 4.5 degrees. The pitch component was also sinusoidal with a frequency of 0.12 Hz and amplitude of 0.3 degrees.

To evaluate the roll angle performance, we manually moved the base at a frequency of 0.1 Hz and an amplitude of five degrees along the roll axis. The roll angle response is shown in the upper plot in Figure 33. From the figure, it can be seen that the PID controller for the roll axis compensated for the ± 5 degree input ship motion and stabilized the roll angle to within ± 1.5 degrees of the horizontal as desired. Furthermore, the average angular error over a ten-second period was approximately 0.2 to 0.3 degrees. We also note that the motor backlash and the elasticity of the 3D-printed structure were significant components of the angular error. As discussed in Chapter II, Section A, a CIWS is sufficiently accurate when the amplitude of the angular error is less than 2.2 degrees. Keeping this in mind and that a CIWS is built to continuously fire against a target for several seconds, the average measured roll error indicates that the stabilizer is suitably accurate for its designed purpose.

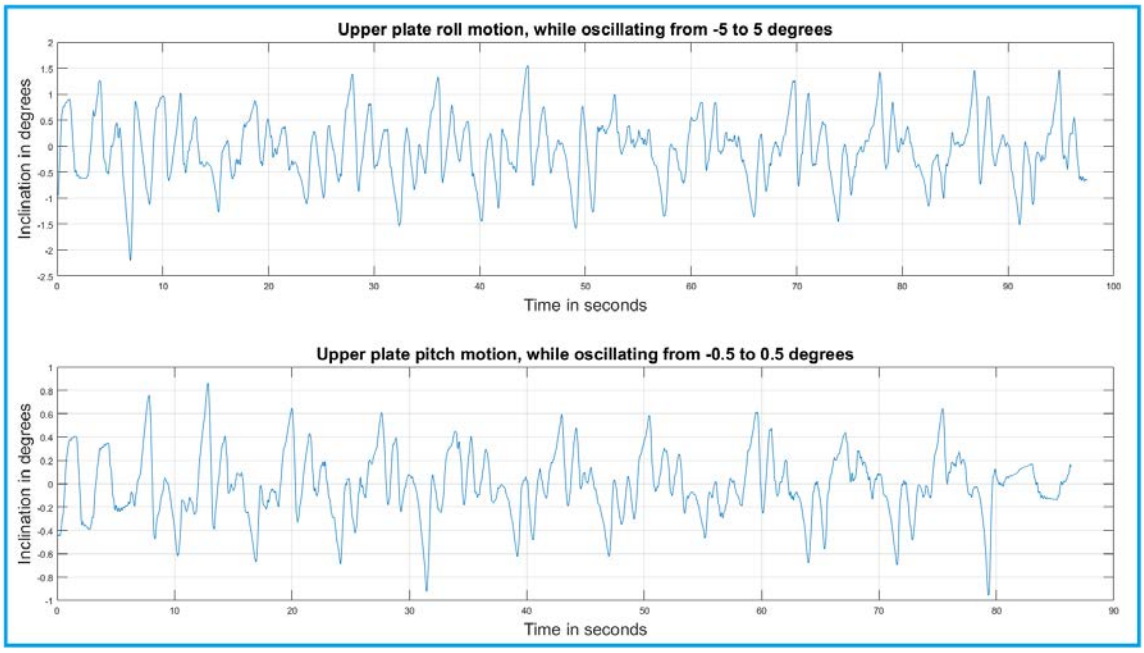

Figure 33. Measured Inclinations of Stabilizer's Upper Plate

In Figure 33, the pitch angle response is shown in the lower plot. To simulate the natural pitch angle motion of the ship, the stabilizer platform was rotated back and forth along the pitch axis at a frequency of 0.1 Hz and an amplitude of 0.5 degrees. As observed in this plot, the PID controller for the pitch axis did not compensate for the ± 0.5 degrees

of sinusoidal movement of the base. Conversely, we observed a pitch angular error of ± 0.8 degrees. Furthermore, we observed similar pitch errors even when the stabilizer base was completely stationary. Consequently, the servomotor backlash and elasticity of the 3D-printed structure were more than likely responsible for these angular errors. To investigate the pitch and roll angle performance a bit more, we conducted several additional experiments with simulated ship motion of different amplitudes but with the same frequency. We observed that the stabilizer did not compensate for ship motions with amplitudes less than one degree due to these two negative factors.

Next, we examined the step response of the stabilizer in each of the axes. Each axis was tested independently one at a time. To begin each measurement, the stabilizer was inclined five degrees in either roll or pitch with the power turned off. Then, when the stabilizer power was turned on and the PID controller program started, the angle was immediately corrected and reduced to close to the desired inclination of zero degrees. In this manner the step response for each axis was measured. Then we plotted the results to visualize these two step responses, as shown in Figure 34. The five-degree initial angular position was selected to keep our step response analysis within the linear area.

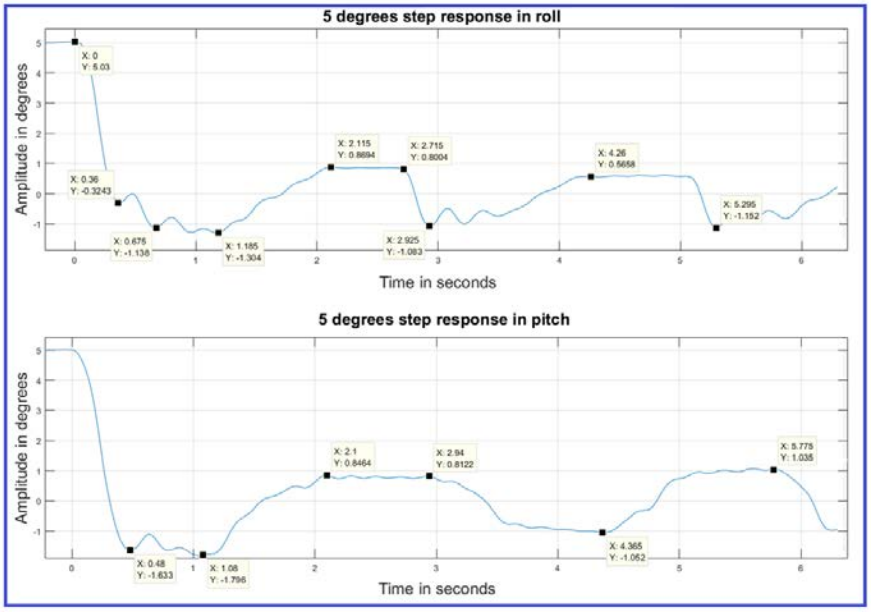

Figure 34. Stabilizer's Five-degree Real Step Response in Roll and Pitch

Analyzing the step response graph for the roll axis, we observe that the upper plate inclination was decreased from five-degrees to zero within 0.315 s. The roll angle error is observed to wander approximately 1.2 degrees about the zero reference. This behavior remains even when the stabilizer's base is stationary after the initial step response has finished. As reported earlier, this error is attributed to the elasticity of the 3D-printed structure and servomotor backlash. We observed similar behavior for the pitch axis. Later in this chapter, we use the two step responses presented here to develop a mathematical model for the stabilizer.

2. Gun Control and Computer Unit Performance

We evaluated the gun control and computer unit performance in two stages. First, we examined the unit's performance when operating against a target moving in the opposite direction, as shown in Figure 35. Through this measurement, we estimated the unit's accuracy while operating onboard a ship. Second, we examined the step response in each of the elevation and azimuth axes. This measurement was used later to develop a mathematical model of the prototype for use in simulation.

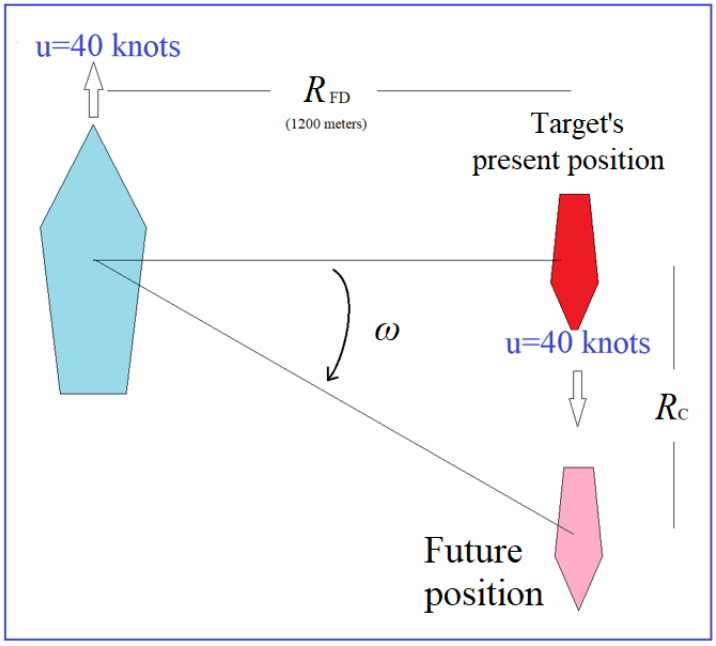

Figure 35. Angular Velocity of an Opposite-moving Target

To simulate the motion of a target, we estimated the relative angular speed ω between our ship that is moving at a speed u and a target traveling at the same speed u in the opposite direction. Given that a typical maximum speed for warships is 40 knots and that the nominal firing distance is 1,200 m, the relative angular velocity is given by

$$\omega = \frac{\arctan\left(\frac{R_C}{R_{FD}}\right)}{T}\ , \tag{11}$$

where R_C is the target's relative distance traveled during a specific time interval T given by

$$R_C = 2uT\ . \tag{12}$$

Substituting Eq. (12) into Eq. (11), we get that a typical relative angular velocity of a target is 3.92 degrees/s.

We then used this last result to create a target that was moving at the aforementioned relative angular velocity with respect to the camera of the gun control unit. While tracking this target, we monitored the system's performance and plotted its response for the azimuth and elevation axes, as shown in Figure 36. Through these plots, we observed that the tracking error slightly exceeded the error limits we had set at the beginning of the design process. Nonetheless, the unit maintained the desired accuracy after the first two seconds in spite of the backlash and elasticity, which created an error of 0.8 degrees, as expected.

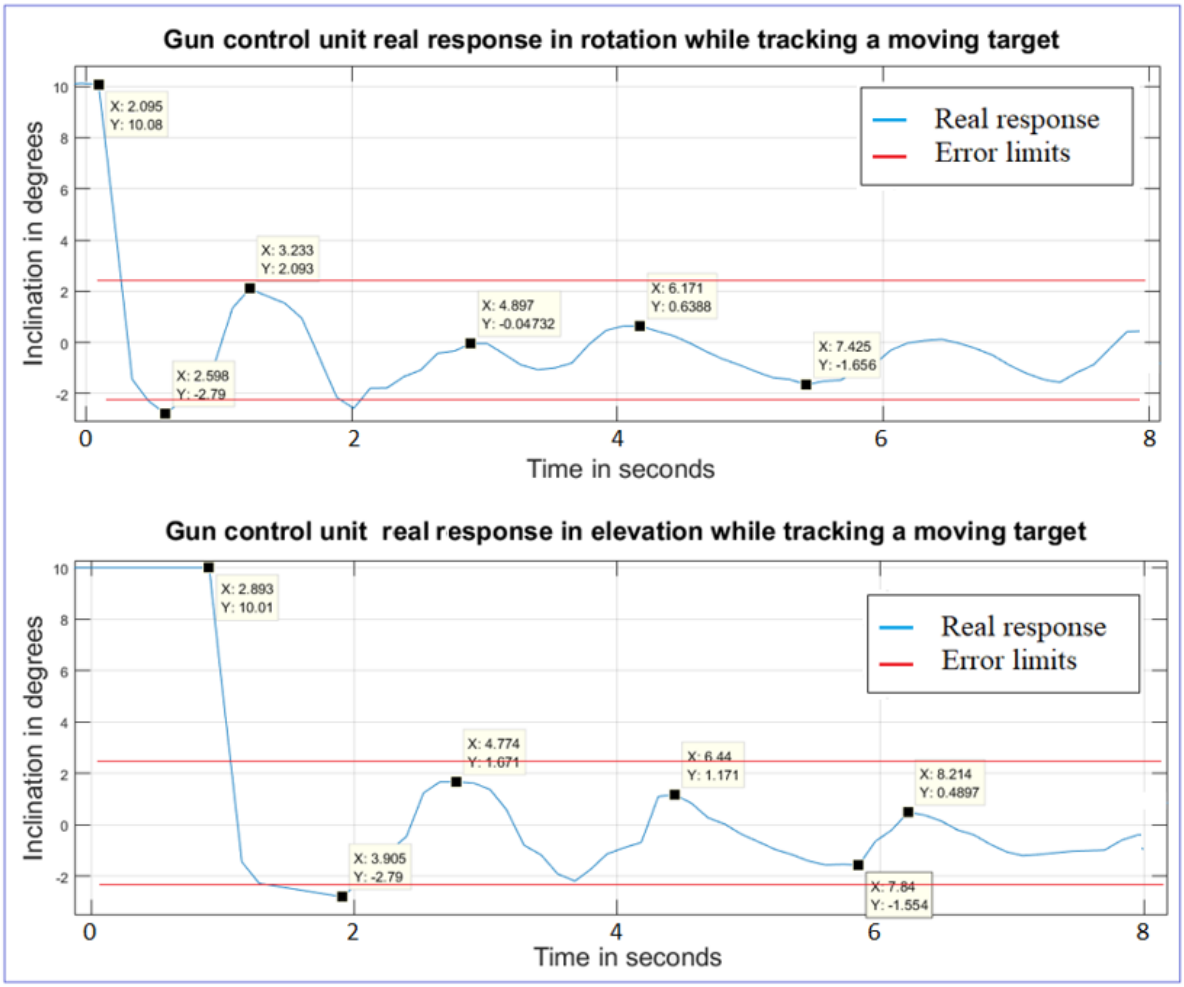

Figure 36. Gun Control Unit Real Response while Tracking a Moving Target

To evaluate the five-degree step response of the system, we followed the same measurement process we used for the stabilizer. The results for each of the axes are shown in Figure 37. Because we use the same type of servomotors that we used for the stabilizer, we observed that they had a similar step response. Finally, we use these results to mathematically model the gun control unit in Section B.

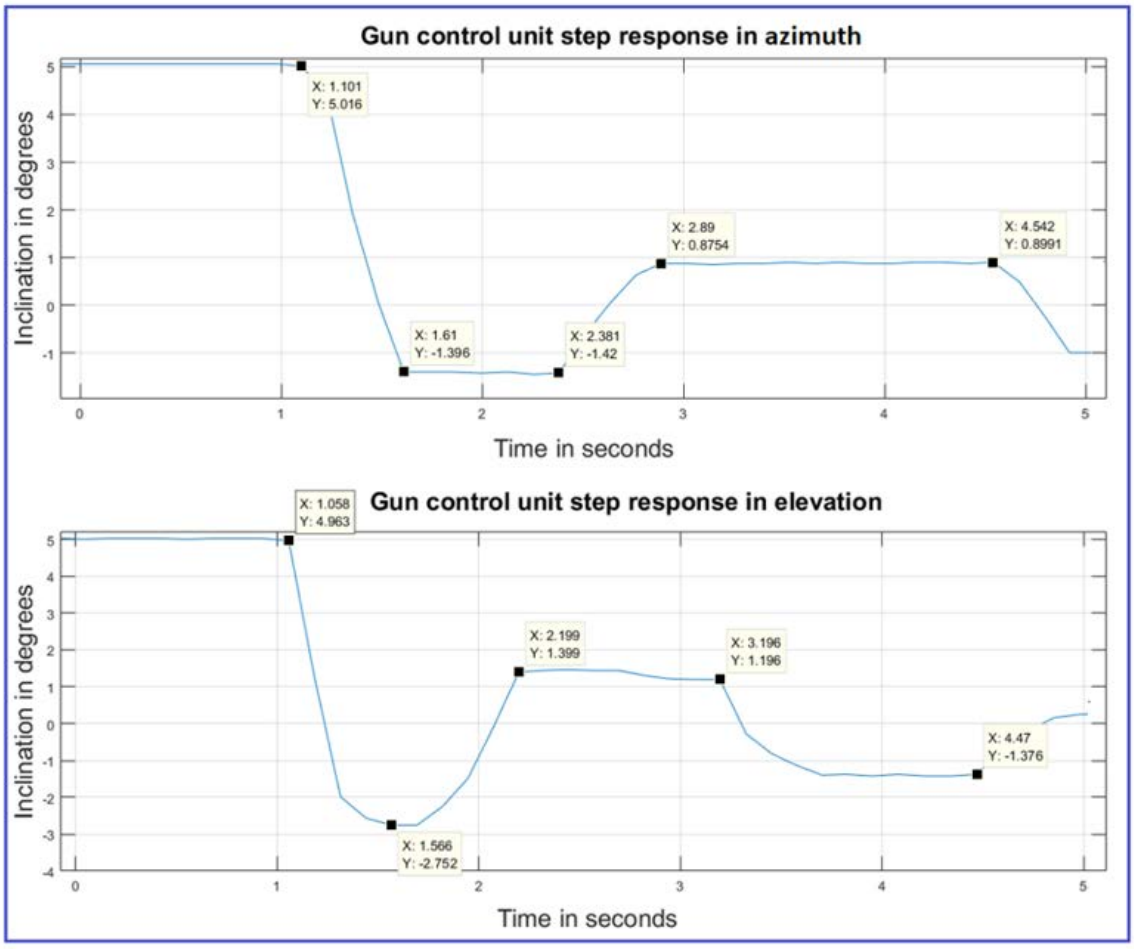

Figure 37. Gun Control Unit Step Response

3. Prototype Overall Performance

As a result of the previous experiments, the IMU sensor began to fail and report faulty measurements. This problem, in combination with the existing structure's elasticity and servomotor backlash, caused the performance of the prototype to degrade. After repeated trials, the 3D-printed structure gradually began to weaken, and some of the metal parts on the servomotor cracked, requiring repair. As a result, the overall performance of the prototype significantly decreased and highlighted the need for some parts of the prototype to be redesigned. Nevertheless, our previous measurements were sufficient to provide enough data to develop a mathematical model of the system and to create a simulation in Simulink which could be used for additional performance analysis.

B. SIMULATION OF THE CIWS

The simulation aimed to determine the limits of the prototype's performance. It provided a means to further examine the response of the prototype CIWS and to study the response for those inputs that cannot be easily reproduced in the laboratory without specialized test equipment. By modeling the system in Simulink, we examined the system's performance for different ship motions and relative target motions. First, we developed a mathematical model of the system using the measured step response presented earlier in the chapter. Then, we created a model in Simulink that operated similarly to the prototype, and we compared its response with the real step responses. We also verified the sinusoidal motion response, as well, to convince ourselves of the accuracy of our model. Finally, the model was used to extrapolate the performance of the CIWS prototype for different dynamic ship motions.

1. Stabilizer Mathematical Model and Simulation

Taking into consideration the measured step response, we created a mathematical model that described the stabilizer's behavior. We created a second-order transfer function in Simulink with a feedback loop for each of the axes, as shown in Figure 38. Through trial and error, we found a transfer function that produced a step response similar to the stabilizer's real step response in the roll axis. This transfer function was found to be

$$\frac{55}{s^2 + 3.2s} \ . \tag{13}$$

Plots of the real and the simulated step response are shown in Figure 39. We observed that they were almost identical during the first 0.3 s. Similarly, we determined the pitch axis transfer function to be

$$\frac{55}{s^2 + 5.4s} \ . \tag{14}$$

We also plotted the simulated and the real step response of the pitch axis, as shown in Figure 40.

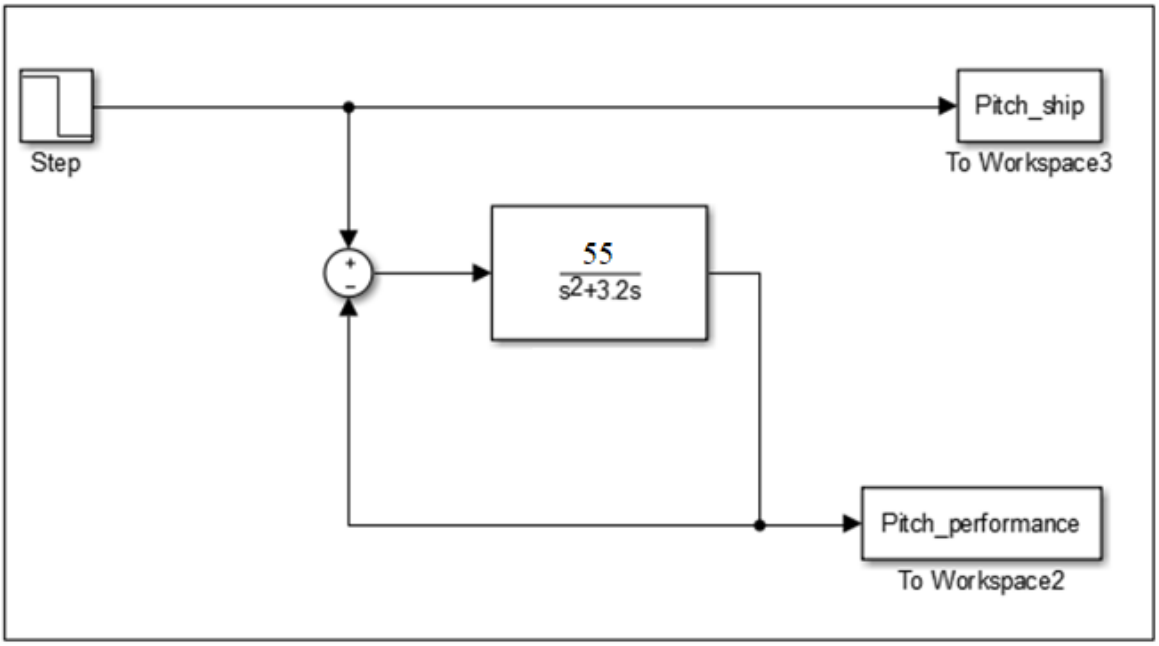

Figure 38. Stabilizer's Single Axis Simulation Model

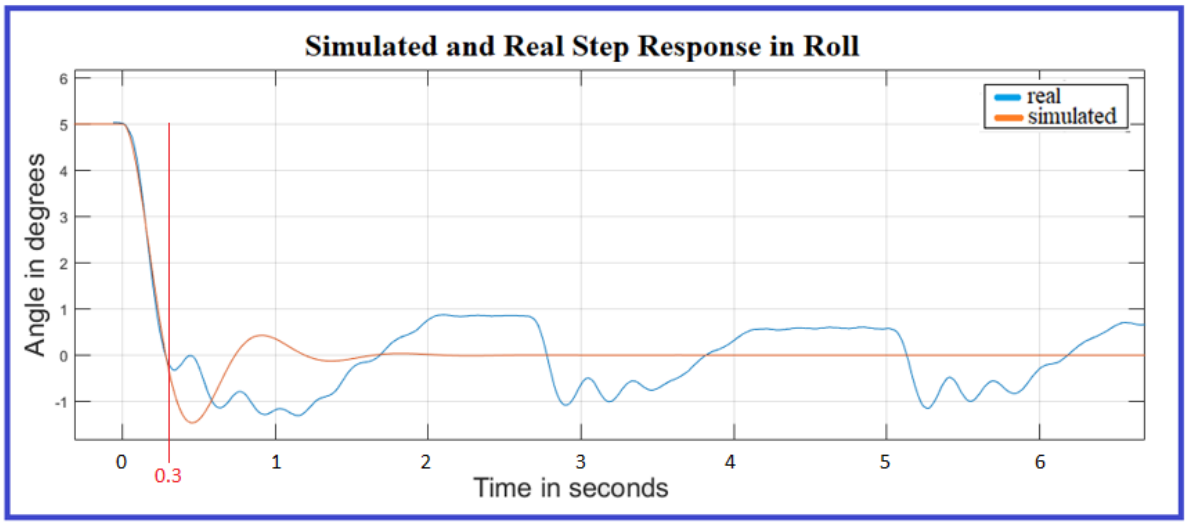

Figure 39. Stabilizer's Simulated and Real Step Response in Roll

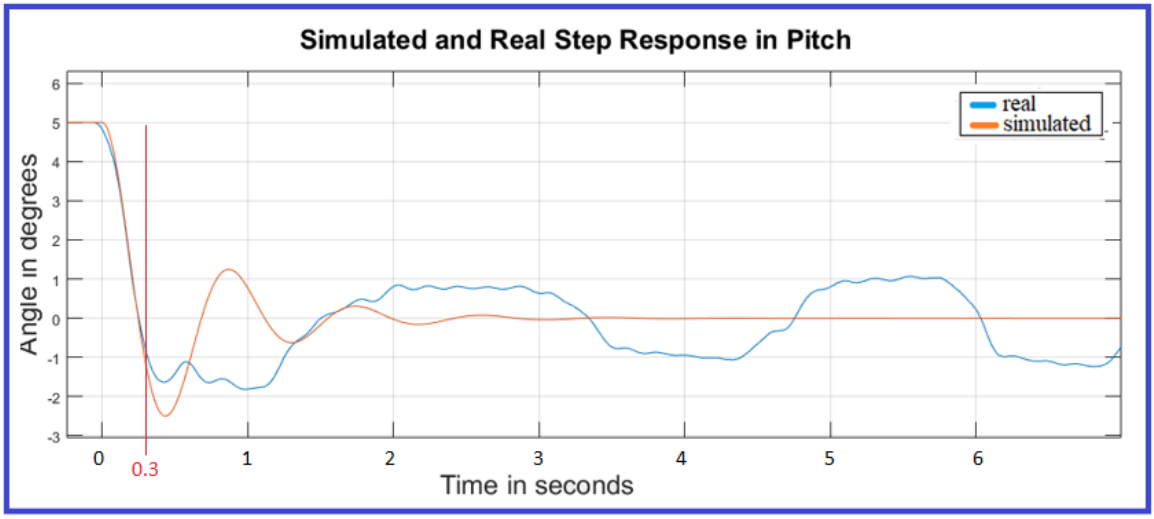

Figure 40. Stabilizer's Simulated and Step Response in Pitch

The overall Simulink model of the stabilizer is shown in Figure 41. For each of the axes of our model, we included a random number generator to act as a disturbance representing the stabilizer's elasticity and backlash errors. Then, we applied as an input a sinusoidal motion at a frequency of 0.1 Hz and amplitude five degrees, as we did previously for the actual prototype. We compared their responses, and we plotted the results in each of the axes, as shown in Figure 42. As expected, the simulated and real plots have similar frequencies and amplitudes. Moreover, their average stabilizing errors appeared almost the same. With these results, we concluded that our simulated model had similar performance characteristics to the actual prototype.

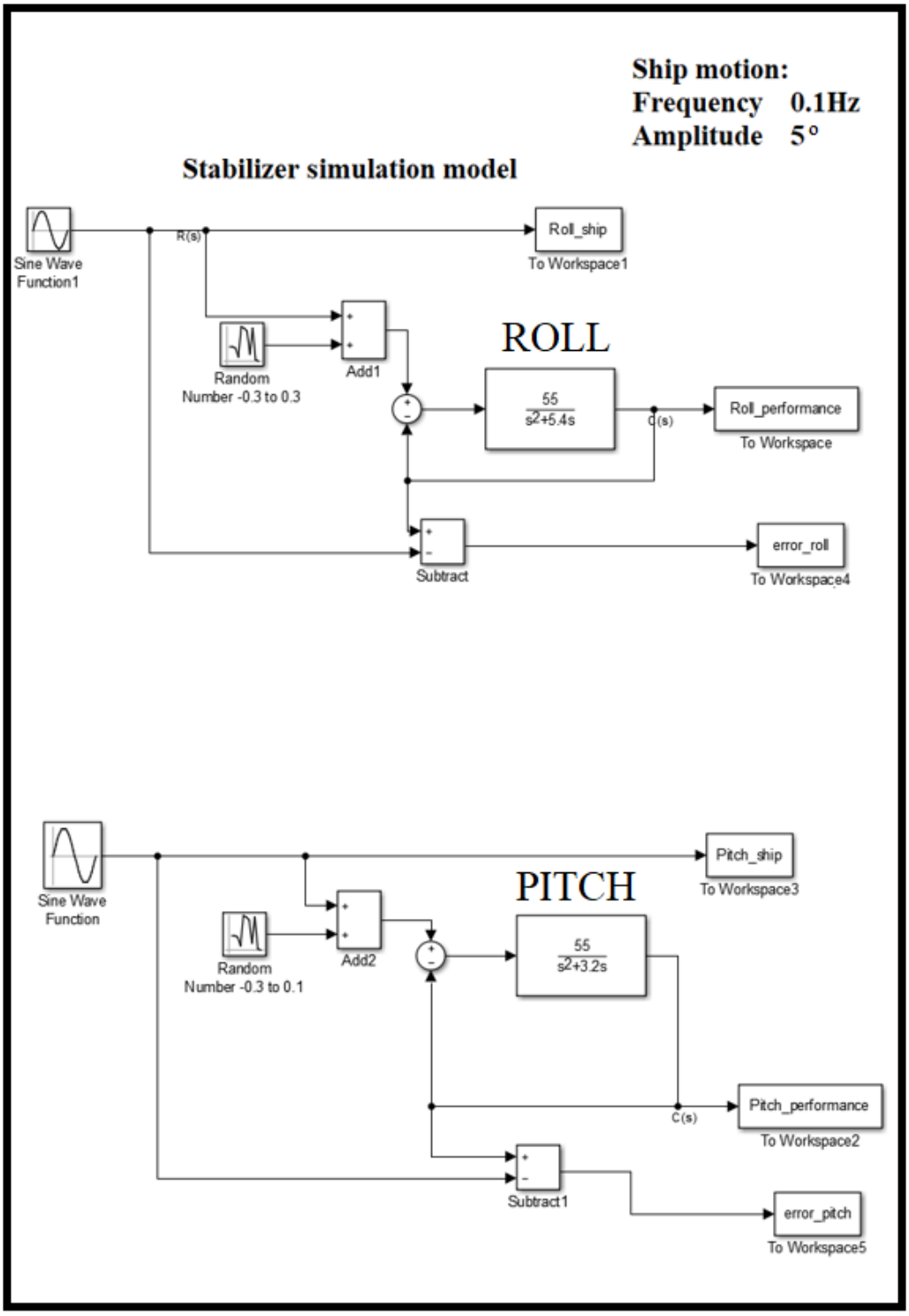

Figure 41. Stabilizer's Simulink Model

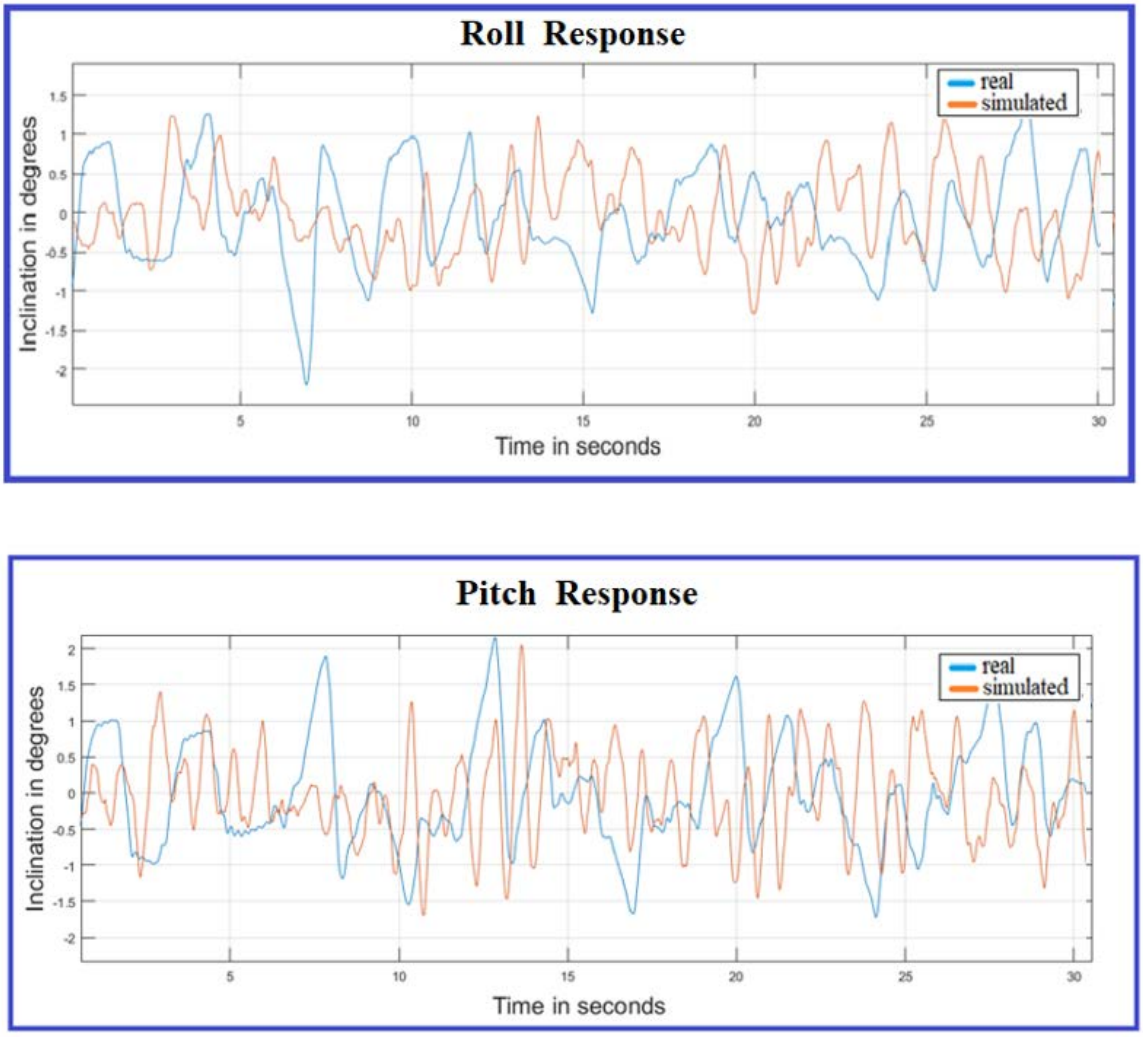

Figure 42. Simulated and Real Response in Each of the Axes

Since we were satisfied with the operation of our model simulation of the stabilizer, we used it to predict the performance of the actual prototype for different dynamic motion inputs. Through these additional simulations, we observed that if the frequency of the motion was 0.5 Hz or more, the amplitude of the stabilizing error exceeded 2.2 degrees, as shown in Figure 43. This frequency corresponded to motion having a period of two seconds, which is typical for smaller boats. In Chapter II, we found that angular errors exceeding this range did not satisfy the performance requirements for the prototype CIWS.

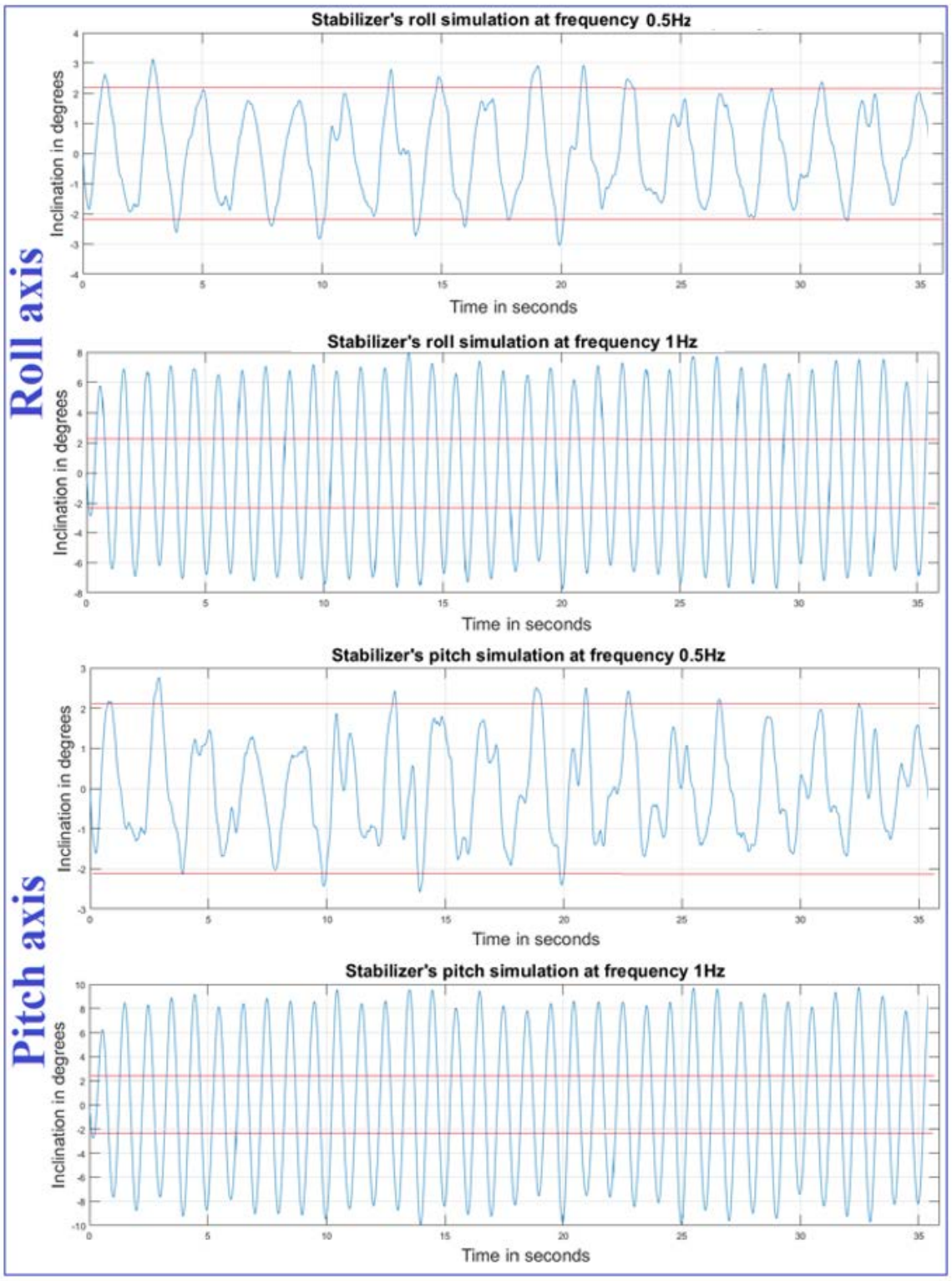

Figure 43. Stabilizer's Performance in Different Frequencies

2. Gun Control and Computer Unit Mathematical Model and Simulation

To model the gun control unit, we followed the same process as for the stabilizer. After some trial and error, we determined the transfer function for the azimuth angle to be

$$\frac{70}{s^2+5s}\ . \tag{15}$$

The transfer function for the elevation angle was found to be

$$\frac{120}{s^2+4s}. \tag{16}$$

Using these two transfer functions, we created a Simulink model for each of the elevation and azimuth axes. The real and the simulated step response were almost identical for the first 0.25 s, as shown in Figure 44.

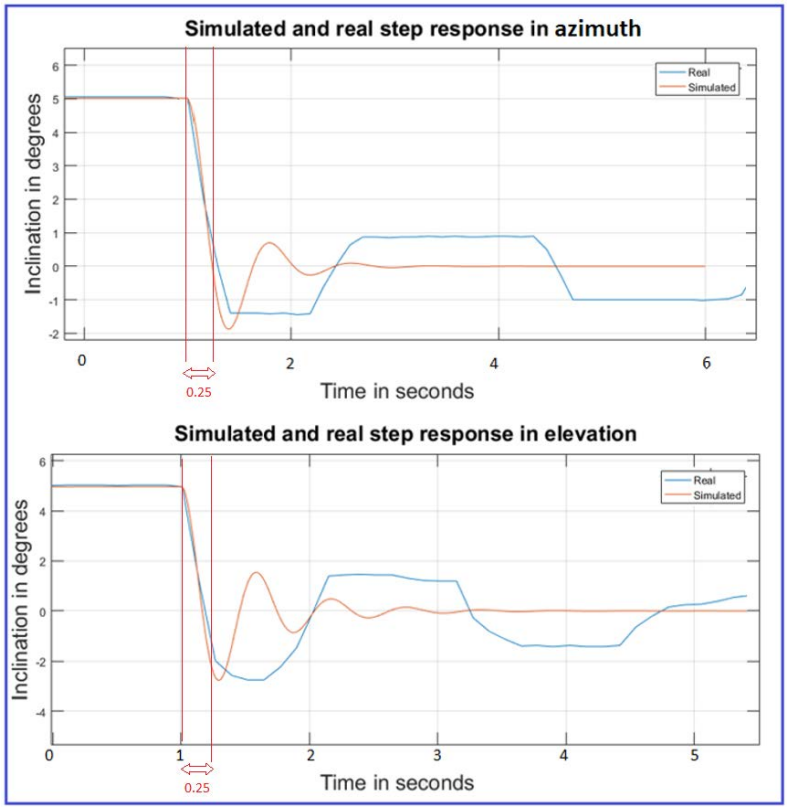

Figure 44. Real and Simulated Gun Control Unit Step Response

The overall Simulink model of the unit is shown in Figure 45. As we did with the stabilizer, we also included a disturbance generator. Then, we applied a constant input to the model to represent the moving target that moves at a constant angular speed of 2.2 degrees/s. We compared the real with the simulated response, and we plotted the results for each of the axes, as shown in Figure 46. Through trials we observed that the gun control unit operated similarly to the aforementioned tracking plot for every moving target with a constant speed; however, the angular speed of the target must be less than the operational limits of the motors, 120 degrees/s.

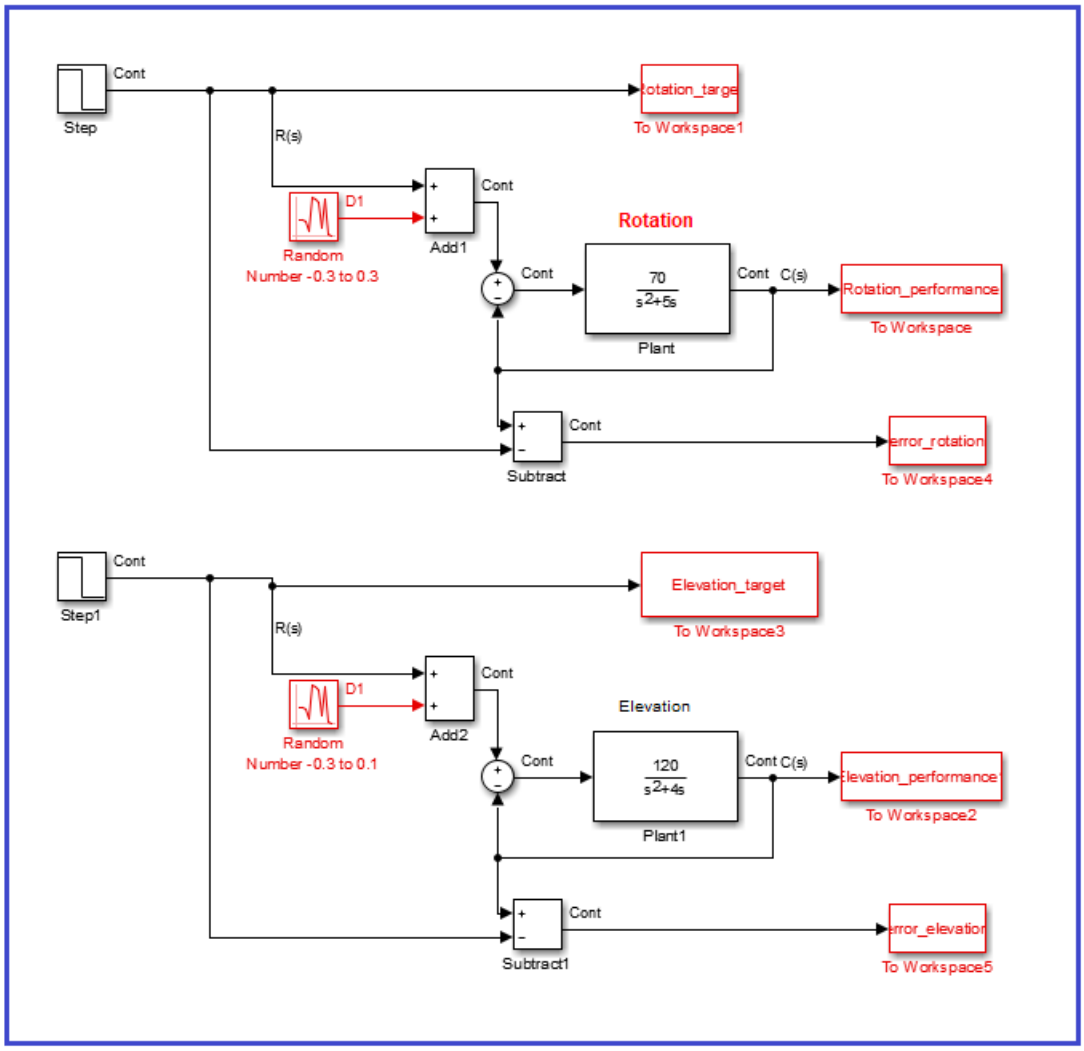

Figure 45. Gun Control Unit Simulink Model

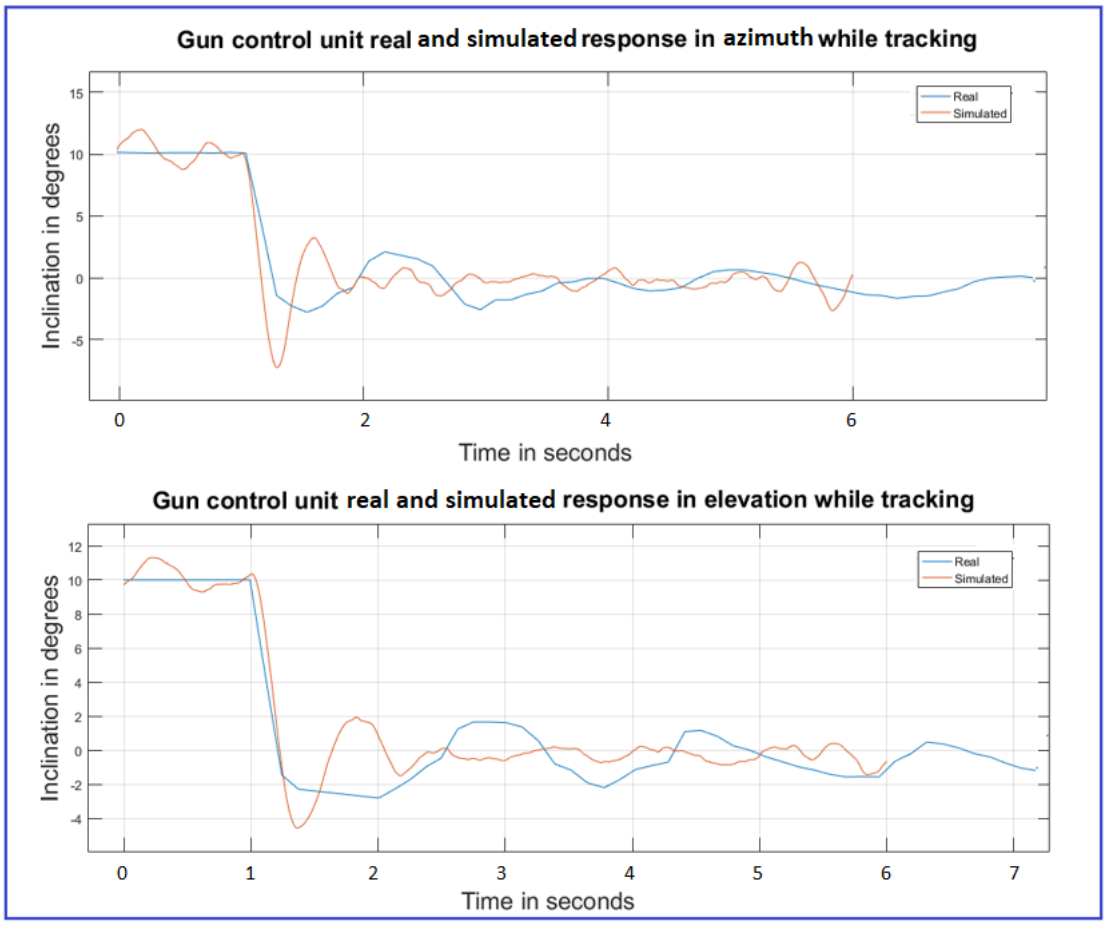

Figure 46. Real and Simulated Response while Tracking

V. CONCLUSIONS

In this thesis research, we investigated the feasibility of creating a portable CIWS that can be easily transferred and installed on small warships. Specifically, the objective was to design and build a prototype CIWS using commercial off-the-shelf components. The prototype that was developed used the popular Arduino microcontroller, the MPU-6050 IMU, a webcam, and four DC servomotors. A supporting structure made from 3D printed parts was used to mount all of the hardware. Software was developed for the Arduino microcontrollers to implement the needed PID controllers. Additional software algorithms were developed in the Processing programming language for the image processing task and to implement the automated target tracking capability of the prototype CIWS.

To characterize the operating environment of the prototype CIWS, pitch and roll measurements were acquired aboard a Hellenic navy ship. These measurements were then used to develop the performance specification for the CIWS. In addition, open video sources were analyzed to estimate the tracking accuracy of existing CIWS and for comparison with our prototype.

Experimental measurements and simulations were conducted to determine the performance of the CIWS prototype. The results indicated that the tracking accuracy of the prototype was comparable to that of existing CIWS. With regard to the stabilizer, the stabilizing error was less than two degrees in ship motions at frequencies less than 0.5 Hz. On the other hand, the overall tracking error of the system was found to be less than one degree.

To develop the CIWS beyond the initial prototype described in this research, several modifications are required. First, the supporting structure, which tended to flex because of the 3D printed material that was used, needs to be replaced with a more rigid material that is durable and able to operate in a naval environment. The supporting structure's lack of rigidity negatively affected the operation of the PID controllers. Another required improvement is to replace the DC servomotors with units that are more suited to

the application. The units that were selected for the prototype were chosen for their low cost and light weight. The DC servomotors, however, exhibited considerable gear backlash that contributed to the overall performance error. To improve this, we recommend using a higher-quality gearbox with less backlash. Another recommended improvement is either to replace the MPU-6050 IMU with a more reliable and accurate sensor or, as suggested in other sources [19], to implement a combination of IMU sensors to achieve greater accuracy. Finally, the integration of a range measurement device, such as a LIDAR, would be another improvement useful to determine the range of a moving target and predict its future position sequentially.

APPENDIX A. ARDUINO CODE FOR THE STABILIZER

```
#include "I2Cdev.h"
#include <Servo.h>
#include <Wire.h>
Servo servoX;
Servo servoY;
int apotelesmata;
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
double time,timePrev;
double elapsedTime,dt;
double Setpoint, Input, Output;
float mpuy,mpux;
float Xx=80,Yy=88,oldmpux=1,oldmpuy=1;
int ssincomingByte ;
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

// ===               INTERRUPT DETECTION ROUTINE                ===
volatile bool mpuInterrupt = false;     high
void dmpDataReady() {
   mpuInterrupt = true;
}
// ===                      INITIAL SETUP                       ===
void setup() {
   time = millis();
```

```
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif
    // initialize serial communication
    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration, others continue immediately
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));
    Serial.println(F("\nSend any character to begin DMP programming and demo: "));
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788);
    if (devStatus == 0) {
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

servoX.attach(5,500,2500);
servoY.attach(6,250,2500);
servoX.write(Xx);
servoY.write(Yy);
delay(2000);
 }

// ===                MAIN PROGRAM LOOP                ===
```

```
void loop()
  {
    time = millis();
    dt=time-timePrev;
    timePrev=time;
    Setpoint=0;
    if (!dmpReady) return;
    while (!mpuInterrupt && fifoCount < packetSize) {    }
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();
    fifoCount = mpu.getFIFOCount();
    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
       // reset so we can continue cleanly
       mpu.resetFIFO();
       Serial.println(F("FIFO overflow!"));

    } else if (mpuIntStatus & 0x02) {
       // wait for correct available data length, should be a VERY short wait
       while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    // read a packet from FIFO
       mpu.getFIFOBytes(fifoBuffer, packetSize);
       // track FIFO count here in case there is > 1 packet available
       // (this lets us immediately read more without waiting for an interrupt)
       fifoCount -= packetSize;

      #ifdef OUTPUT_READABLE_QUATERNION
          // display quaternion values in easy matrix form: w x y z
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          Serial.print("quat\t");
          Serial.print(q.w);
          Serial.print("\t");
          Serial.print(q.x);
          Serial.print("\t");
          Serial.print(q.y);
          Serial.print("\t");
          Serial.println(q.z);
      #endif

      #ifdef OUTPUT_READABLE_EULER
          // display Euler angles in degrees
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          mpu.dmpGetEuler(euler, &q);
          Serial.print("euler\t");
          Serial.print(euler[0] * 180/M_PI);
```

```
            Serial.print("\t");
            Serial.print(euler[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(euler[2] * 180/M_PI);
        #endif

        #ifdef OUTPUT_READABLE_YAWPITCHROLL
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
//--------------------------------- Debugging in the serial monitor-----------------------
            //Serial.print(apotelesmata);
            //Serial.print("ypr\t");
            //Serial.print(ypr[0] * 180/M_PI);
            //Serial.print("\t");
             Serial.println(ypr[1] * 180/M_PI);
            // Serial.print("\t");
            //Serial.println(ypr[2] * 180/M_PI);
            Serial.print("\t");
            //Serial.println(elapsedTime);
//--------------------------------- Results in degrees from mpu-6050 ----------------------
          mpux = ypr[1] * 180/M_PI;       // ready to use value from mpu-6050 in x-axis
          mpuy = ypr[2] * 180/M_PI;
//------------------PD controller for the x-axis & y-axis stabilizer-servo motors------
  apotelesmata ++;
  if ((Xx==80) &&(Yy==88)&&(apotelesmata<1500))
          {                           // servos looked in ... degrees until the mpu is stabillized
            Xx=80;
            Yy=88;
          }
  else
         {   //====================    X-AXIS ======================
            Xx= Xx -0.10*mpux -18*(mpux-oldmpux)/dt;
            oldmpux=mpux;
            //====================   Y-AXIS ======================
            Yy= Yy -0.05*mpuy -11*(mpuy-oldmpuy)/dt;
            oldmpuy=mpuy;
          }
  if (Yy>120)Yy=115;
  else if (Yy<60) Yy=65;
  if (Xx>120)Xx=115;
  else if (Xx<60) Xx=65;
  servoX.write(Xx);
  servoY.write(Yy);
```

```
        #endif
        #ifdef OUTPUT_READABLE_REALACCEL
            // display real acceleration, adjusted to remove gravity
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetAccel(&aa, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
            Serial.print("areal\t");
            Serial.print(aaReal.x);
            Serial.print("\t");
            Serial.print(aaReal.y);
            Serial.print("\t");
            Serial.println(aaReal.z);
        #endif
        #ifdef OUTPUT_READABLE_WORLDACCEL
            // display initial world-frame acceleration, adjusted to remove gravity
            // and rotated based on known orientation from quaternion
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetAccel(&aa, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
            mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
            Serial.print("aworld\t");
            Serial.print(aaWorld.x);
            Serial.print("\t");
            Serial.print(aaWorld.y);
            Serial.print("\t");
            Serial.println(aaWorld.z);
        #endif
        #ifdef OUTPUT_TEAPOT
            // display quaternion values in InvenSense Teapot demo format:
            teapotPacket[2] = fifoBuffer[0];
            teapotPacket[3] = fifoBuffer[1];
            teapotPacket[4] = fifoBuffer[4];
            teapotPacket[5] = fifoBuffer[5];
            teapotPacket[6] = fifoBuffer[8];
            teapotPacket[7] = fifoBuffer[9];
            teapotPacket[8] = fifoBuffer[12];
            teapotPacket[9] = fifoBuffer[13];
            Serial.write(teapotPacket, 14);
            teapotPacket[11]++; // packetCount, loops at 0xFF on purpose
        #endif

    }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. ARDUINO CODE FOR THE GUN CONTROL UNIT

```
#include <Servo.h>
#include <LiquidCrystal.h>
Servo myservo1;
Servo myservo2;
int buffer [16];
float angleX = 78;
float angleY = 78;
int flag=0;

void setup() {
  Serial.begin(115200);
  myservo1.attach(8);     //  azimuth
  myservo2.attach(9);     // Elevation
  myservo1.write(angleX);
  myservo2.write(angleY);
  delay(1000);
}

void loop() {
  if (Serial.available() > 0) {
    if (Serial.read() == '$') {
      for (int i = 0; i < 12; i++) {
        buffer[i] = Serial.read() - '0';
        delay(3);
      }
      angleX = (buffer[1] * 1000 + buffer[2] * 100 + buffer[3]*10 + buffer[4])/10;
      angleY = (buffer[7] * 1000 + buffer[8] * 100 + buffer[9]*10 + buffer[10])/10;
      Serial.flush();
    }
      if (angleX<50) angleX=50;
      else if(angleX>140) angleX=140;
      if (angleY<50) angleY=50;
      else if(angleY>140) angleY=140;
      flag=1;    //starting control from processing program
      myservo1.write(angleX);
      myservo2.write(angleY);

  }
     }
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. CV CODE FOR THE COMPUTER UNIT

```
import processing.video.*;
import processing.serial.*;
Capture video;
Serial myPort;  // Create object from Serial class
int motor=0;
color trackColor=1;
float threshold = 35;             // color threshold
float distThreshold = 50;         // distance between two blobs
float Ltargetsize= 6500;          // Largest target  that detects
float Stargetsize= 30;            // Smaller target that detects
float errorX=0;
float p_errorX=0;
float errorY=0;
float p_errorY=0;
float a=0;
float angleXlim=0;
float angleYlim=0;
int i=0;
int j=0;
int cc=3;
int bb=3;
int startTime;
float oldtime=0;
float newtime=0;
float dt=0;
float timecounter=0;
ArrayList<Blob> blobs = new ArrayList<Blob>();
boolean recording = false;
boolean trackingON = false;
boolean red_target_window=false;

// arxikes times strofis kai ipsosis
float angleX=78;                  // initial angle of servo1
float angleY=78;                  // initial angle of servo2

// boundaries in elevation azimuth
int KorioY = 60;
int PorioY = 100;
int KorioS = 50;
int PorioS = 120;

//improvement while auto tracking
```

```
int opitimize_trackingX=0;
int opitimize_trackingY=0;

// initial scope size
float scopeSize=450;
void setup() {
  size(1600, 896);
  startTime = millis();
  String portName = Serial.list()[0]; // change the 0 to a 1 or 2 etc. to match your port
  myPort = new Serial(this, portName, 115200);   //arduino connection
  String[] cameras = Capture.list();
  printArray(cameras);
  //video = new Capture(this, 1920, 1080);
  /*    Some of the modes for the existing camera
 [101] "name=Logitech HD Pro Webcam C920,size=1600x896,fps=30"
 [119] "name=Logitech HD Pro Webcam C920,size=640x360,fps=30"
 [121] "name=Logitech HD Pro Webcam C920,size=800x448,fps=30"
 [123] "name=Logitech HD Pro Webcam C920,size=800x600,fps=30"
 [125] "name=Logitech HD Pro Webcam C920,size=864x480,fps=30"
 [127] "name=Logitech HD Pro Webcam C920,size=960x720,fps=30"
 [129] "name=Logitech HD Pro Webcam C920,size=1024x576,fps=30"
 [131] "name=Logitech HD Pro Webcam C920,size=1280x720,fps=30"
 [133] "name=Logitech HD Pro Webcam C920,size=1600x896,fps=30"
 */
  video = new Capture (this, Capture.list()[101]);          // choose number from printed list
below the program
  video.start();
  trackColor = color(255, 0, 0);
}

void captureEvent(Capture video) {
  video.read();
}

// improvements and threasholds with keyboard
void keyPressed() {
  if (key == 'a') {
    distThreshold+=5;
  } else if (key == 'z') {
    distThreshold-=5;
  }
  if (key == 's') {
    threshold+=5;
  } else if (key == 'x') {
    threshold-=5;
```

```
}
if (key == 'd') {
  Ltargetsize+=20;
} else if (key == 'c') {
  Ltargetsize-=20;
}
if (key == 'f') {
  Stargetsize+=20;
} else if (key == 'v') {
  Stargetsize-=20;
}
if (key == 'r' || key == 'R') { // video  recording
  recording =! recording;
}
if (key == 'b') {
  scopeSize+=20;
} else if (key == 'g') {
  scopeSize-=20;
}
// use like joistic     but differs while auto tracking  OPTIMIZE
if ((keyCode == UP) && (angleY<PorioY)) {
  if (trackingON) {
    opitimize_trackingY+=5;
  } else {
    angleY+=0.5;
  }
} else if ((keyCode == DOWN) && (angleY>KorioY)) {
  if (trackingON) {
    opitimize_trackingY-=5;
  } else {
    angleY-=0.5;
  }
}
if ((keyCode == LEFT) && (angleX>KorioS)) {
  if (trackingON) {
    opitimize_trackingX-=5;
  } else {
    angleX-=0.5;
  }
} else if ((keyCode == RIGHT) && (angleX<PorioS)) {
  if (trackingON) {
    opitimize_trackingX+=5;
  } else {
    angleX+=0.5;
  }
```

```
  }
  if (key == 'P' || key == 'p') { // give or take the control to auto-traking
    trackingON =! trackingON;
    if (!trackingON) scopeSize=450;
    opitimize_trackingY=0;
    opitimize_trackingX=0;
    angleXlim=angleX;
    angleYlim=angleY;
    i=0;
    j=0;
  }
}

void draw() {
  video.loadPixels();
  image(video, 0, 0);
  int elapsed = millis() - startTime;
  oldtime=newtime;
  newtime= float(elapsed) / 1000;
  dt=newtime-oldtime;
  //timecounter=timecounter+dt;
  //println(dt);
  blobs.clear();  // blobs are areas with pixels having color under the color threashold
  // Begin loop to walk through every pixel
  for (int x = 0; x < video.width; x++ ) {
    for (int y = 0; y < video.height; y++ ) {
      int loc = x + y * video.width;
      // What is current color
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);

      float d = distSq(r1, g1, b1, r2, g2, b2);  // color distance

      if (d < threshold*threshold) {

        boolean found = false;
        for (Blob b : blobs) {
          if (b.isNear(x, y)) {
            b.add(x, y);
            found = true;
```

```
          break;
        }
       }

       if (!found) {
        Blob b = new Blob(x, y);
        blobs.add(b);
       }
      }
     }
   }
    for (Blob b : blobs) {
    if ((b.size() > Stargetsize) && (b.size() < Ltargetsize)) {   // rejects targets smaller
      //  print the center of the target on screen
      float [] delta=b.show();    // delta[0] is the centerX of target and  delta[1] the centerY
      // follows whatever is inside the red scope.
      if        ((delta[0]>(scopeSize))        &&        (delta[0]<(width-scopeSize))        &&
(delta[1]>(scopeSize/2)) && (delta[1]<(height-scopeSize/2))) {
       fill(0, 255, 0);
       textSize(20);
       text("    Target LOCKED    center   X=" + delta[0], width/2+50, 30);
       text("Y=" + delta[1], width/2+150, 30);
       text("Size=" + b.size(), width/2+50, 50);
       textSize(15);
       text("Camera center       X=" + width/2, width/2, height-25);
       text("Y=" + height/2, width/2+100, height-25);
//println((delta[1]-448)*0.046512,newtime);
  println((delta[0]-800)*0.04731861,newtime);
//1pixel in x = 0.04731861 degrees    800
//1pixel in y = 0.046512   degrees    448

       //===========     PID CONTROL    ========================

       // auto tracking (inside the red box that appears on screen)
       if (trackingON) {
        i++;
        errorX=(delta[0]+opitimize_trackingX-width/2);
        if (i==1) {
          //angleX= map(errorX, -800, 800, angleXlim-26, angleXlim+26);
        } else if (i>8) {
          errorX= map(errorX, -800, 800, -27, +27);
          if (abs(errorX)>0) angleX=angleX + 0.4*errorX + 0.1*(errorX-p_errorX)/dt;
          p_errorX=errorX;
        }
```

```
      //=======================Y-AXIS=====================
      j++;
      errorY=(delta[1]-opitimize_trackingY-height/2);
      if (j==1) {
        //angleY= map(errorY, 448, -448, angleYlim-10, angleYlim+10);
      } else if (j>8) {
        errorY= map(errorY, 448, -448, -10, +10);
        if (abs(errorY)>0) angleY=angleY + 0.4*errorY + 0.1*(errorY-p_errorY)/dt;
        p_errorY=errorY;
      }
      //================================================
      if (i==12) scopeSize=630;
      // Safety-Boundaries in elevation and azimuth
      // when reaches the boundaries must return 1 angle degree back
      if (angleX==PorioS) {
        angleX=PorioS-1;
      } else if (angleX==KorioS) {
        angleX=KorioS+1;
      }

      if (angleY==PorioY) {
        angleY=PorioY-1;
      } else if (angleY==KorioY) {
        angleY=KorioY+1;
      }
    }
  }
 }
}

// Sreen data
rectMode(CENTER);
textSize(12);
rect(50, 80, sqrt(Ltargetsize), sqrt(Ltargetsize));
rect(40, 180, sqrt(Stargetsize), sqrt(Stargetsize));
textAlign(LEFT);
text("Target smaller than: " + Ltargetsize, 10, 15);
text("Target bigger  than: " + Stargetsize, 10, 150);
textAlign(RIGHT);
text("distance threshold: " + distThreshold, width-10, 15);
line(width-20, 30, width-distThreshold, 30);
text("color threshold: " + threshold, width-10, 50);
fill(0);
// skopeftiko
stroke(255, 0, 0);
```

```
 line(width/2-30, height/2, width/2+30, height/2);
 line(width/2, height/2-30, width/2, height/2+30);
 rectMode(CENTER);
 noFill();
 rect(width/2, height/2, width-2*scopeSize, height-scopeSize, 40);

 //record screen
 if (recording) {
  stroke(0, 255, 100);
  fill(0, 255, 0);
  saveFrame("output/gol_####.png");
  textSize(30);
  text("REC", 80, 350);
 }
 if (trackingON) {

  textSize(30);
  stroke(0, 255, 0);
  fill(0, 255, 0);
  text("ON", 80, 270);
 }
 // conection to arduino  throught serial splitting the massage
 String c1="$";
 String c4="0";
 String c5="$07800780";  // initial position
 String c2=str(round(1000*angleX));
 String c3=str(round(10*angleY));

 if (angleX<100 &&angleX>=10) {
  c2=c4+c2;
 } else if (angleX<10) {
  c2=c4+c4+c2;
 }
 if (angleY<100 &&angleY>=10) {
  c3=c4+c3;
 } else if (angleY<10) {
  c3=c4+c4+c3;
 }
 c5=c1+c2+c3;
 myPort.write(c5);
 myPort.clear();
 delay(90);          //     maybe needs more to slow down the commands to arduino
}
 float distSq(float x1, float y1, float x2, float y2) {
 float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1);
```

```
  return d;
}

// Colordistance funcion
  float distSq(float x1, float y1, float z1, float x2, float y2, float z2) {
  float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) +(z2-z1)*(z2-z1);
  return d;
}

void mousePressed() {
  // Save color where the mouse is clicked in trackColor variable
  int loc = mouseX + mouseY*video.width;
  trackColor = video.pixels[loc];
  opitimize_trackingY=0;
  opitimize_trackingX=0;
}

class Blob {
  float minx;
  float miny;
  float maxx;
  float maxy;

  Blob(float x, float y) {
    minx = x;
    miny = y;
    maxx = x;
    maxy = y;
  }

  float [] show() {    // shows a square around the target
    stroke(0, 255, 0);
    noFill();
    strokeWeight(4);
    rectMode(CORNERS);
    rect(minx, miny, maxx, maxy);
    return new float [] {(minx+maxx)/2, (miny+maxy)/2};  //returns the center of the target
  }

    void add(float x, float y) {
    minx = min(minx, x);
    miny = min(miny, y);
    maxx = max(maxx, x);
    maxy = max(maxy, y);
  }
```

```
    float size() {
    return (maxx-minx)*(maxy-miny);
   }
    boolean isNear(float x, float y) {
    float cx = (minx + maxx) / 2;
    float cy = (miny + maxy) / 2;

    float d = distSq(cx, cy, x, y);
    if (d < distThreshold*distThreshold) {
    return true;
    } else {
      return false;
    }
   }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] S. Crowford, *Twenty-first Century Warships: Surface Combatants of Today's Navies.* St. Paul, MN, USA: MBI Pub. Co., 2002.

[2] F. Norman, *The Naval Institute Guide to World Naval Weapons Systems.* Annapolis, MD, USA: Naval Institute Press, 1989.

[3] T. Szulc, "The "last hope": Russian close-in weapon systems," *Military Technology*, vol. 30, no. 11, pp. 75–80, 2006. Accessed September 5, 2017. [Online]. Available: http://libproxy.nps.edu/login?url=https://search.proquest.com/docview/199096135?accountid=12702

[4] E. H. Lundquist, "Modular warship construction," *Naval Forces*, vol. 33, no. 1, Month 2012.

[5] Daily Military Defense & Archive, "The powerful U.S. CIWS in action against poor boat—CIWS live fire exercise," YouTube, May 15, 2015. Accessed January 23, 2017. [Online]. Available: https://www.youtube.com/watch?v=6a_XYaTgG4Y&t=5s

[6] U.S. Navy, "Official website of the United States Navy." Accessed January 24, 2017. [Online]. Available: http://www.navy.mil/view_image.asp?id=66498

[7] Cata Lin, "Russian warship fire at Somali pirates," YouTube, April 27, 2015. Accessed January 24, 2017. [Online]. Available: https://www.youtube.com/watch?v=eozty7Bb29g

[8] "Goalkeeper CIWS Gun System," YouTube, July 7, 2006. Accessed January 24, 2017. [Online]. Available: https://www.youtube.com/watch?v=nY6nm-6eCzM

[9] "Ship motion is divided into six components in the six degrees of freedom," ResearchGate. Accessed March 20, 2017. [Online]. Available: https://www.researchgate.net/figure/Ship-motion-is-divided-into-six-components-in-the-six-degrees-of-freedom_220868905

[10] "Arduino.cc," Arduino. Accessed April 20, 2017. [Online]. Available: https://playground.arduino.cc/Main/MPU-6050#info.

[11] E. Bekir, *Introduction to Modern Navigation Systems.* Singapore: World Scientific Publishing Co Pte Ltd., 2007. [Online]. ProQuest Ebook Central.

[12] N. Murali, "Balancing instructable robot," Instructables, June 21, 2014. [Online]. Available: https://www.instructables.com/id/Balancing-Instructable- Robot/

[13] “ASME-MXB-High-power-high-torque-servo,” Aliexpress. Accessed June 5 2017. [Online]. Available: https://www.aliexpress.com/store/product/ASME-MXB- High-power-high-torque-servo-the-3600-Degree-servo-12V-24V-380kg-cm- 0/325585_32686976772.html

[14] “ASME-MXB-High-power-high-torque-servo,” Ebay. Accessed June 5 2017. [Online]. Available: http://www.ebay.com/itm/ASME-MXB-High-power-high-torque-servo-the-3600-Degree-servo-12V-24V-380kg-cm /302003723970?Hash=item4650d31ec2:g:C0wAAOSwbYZXeSQm

[15] N. S. Nise, *Control Systems Engineering*. Pomona, CA: Wiley, 2015.

[16] D. H. Ballard and C. M. Brown, *Computer Vision*. Upper Saddle River, NJ, USA: Prentice-Hall Inc., 1982.

[17] D. Shiffman, *Learning Processing: A Beginner’s Guide to Programming Images, Animation, and Interaction*. Burlington, MA, USA: Morgan Kauffmann, 2008.

[18] J. Rowberg, “i2cdevlib.” Accessed March 4 2017. [Online]. Available: https://www.i2cdevlib.com/devices/mpu6050#source

[19] J. Cole, “A personal inertial-navigation system based on multiple distributed, nine-degrees-of-freedom, inertial measurement units,” M.S. thesis, Dept. of Comp. and Elec. Eng., NPS, Monterey, CA, 2016. [Online]. Available: http://hdl.handle.net/10945/51727

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California