NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLICATION OF A LEAP MOTION SENSOR FOR IMPROVED DRONE CONTROL

by

Alfredo Belaunde Sara-Lafosse

December 2017

Thesis Advisor: Xiaoping Yun
Second Reader: James Calusdian

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2017	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE
APPLICATION OF A LEAP MOTION SENSOR FOR IMPROVED DRONE CONTROL

5. FUNDING NUMBERS

6. AUTHOR(S) Alfredo Belaunde Sara-Lafosse

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. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Military and civilian drones in use today have been designed to accomplish a wide array of missions. From an engineering perspective, they are very complex systems that incorporate technologies from a wide range of specialized disciplines. Not surprisingly, operating these complex systems requires advanced and extensive training.

This thesis research proposes a simple and intuitive user interface utilizing a Leap Motion sensor that allows a drone operator to exploit his/her skills more intuitively. The Leap Motion sensor tracks the position and orientation of the user's hand(s), which, in turn, controls the motion of a drone. The acquisition and processing of the Leap Motion sensor data were performed using a programming language called Processing. In this research work, an infrared-controlled helicopter and a radio frequency-controlled quadrotor were operated using this interface. For the user interface, several prototype electronic circuits based on Arduino microcontroller boards, as well as Processing programs, were developed and integrated with the Leap Motion sensor. The complete user interface was successfully tested and demonstrated. It was observed that the user interface makes the control of both drone types easier and more intuitive.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Leap Motion sensor, PID control, UAV, drone, Arduino, image processing, control systems	103
	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.

APPLICATION OF A LEAP MOTION SENSOR FOR IMPROVED DRONE CONTROL

Alfredo Belaunde Sara-Lafosse
Lieutenant Commander, Peruvian Navy
B.S., Peruvian Naval Academy, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2017

Approved by: Xiaoping Yun
Thesis Advisor

James Calusdian
Second Reader

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

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Military and civilian drones in use today have been designed to accomplish a wide array of missions. From an engineering perspective, they are very complex systems that incorporate technologies from a wide range of specialized disciplines. Not surprisingly, operating these complex systems requires advanced and extensive training.

This thesis research proposes a simple and intuitive user interface utilizing a Leap Motion sensor that allows a drone operator to exploit his/her skills more intuitively. The Leap Motion sensor tracks the position and orientation of the user's hand(s), which, in turn, controls the motion of a drone. The acquisition and processing of the Leap Motion sensor data were performed using a programming language called Processing. In this research work, an infrared-controlled helicopter and a radio frequency-controlled quadrotor were operated using this interface. For the user interface, several prototype electronic circuits based on Arduino microcontroller boards, as well as Processing programs, were developed and integrated with the Leap Motion sensor. The complete user interface was successfully tested and demonstrated. It was observed that the user interface makes the control of both drone types easier and more intuitive.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. HISTORY AND APPLICATIONS OF UNMANNED AERIAL VEHICLES 1

B. PROBLEM STATEMENT 4

C. THESIS GOALS 4

D. ORGANIZATION 5

II. INPUT DEVICE AND SOFTWARE 7

A. LEAP MOTION SENSOR 7

B. ARDUINO MICROCONTROLLER 10

C. PROCESSING SOFTWARE 11

III. SYSTEM DESIGN AND IMPLEMENTATION 13

A. OPEN-LOOP CONTROL OF THE FQ777 INFRARED-CONTROLLED HELICOPTER 14

1. Introduction 14

2. Implementation 14

3. System Testing 28

B. OPEN-LOOP CONTROL OF THE JJRC H31 QUADROTOR 30

1. Introduction 30

2. Implementation 31

3. System Testing 36

C. CLOSED-LOOP SYSTEM USING WEBCAM AND LEAP MOTION SENSOR 37

1. Introduction 37

2. Implementation 37

3. **System Testing 41**

IV. CONCLUSIONS 45

A. SUMMARY 45

B. FUTURE RESEARCH 46

APPENDIX A. MATRIX OF IR COMMANDS FOR THE FQ777 49

APPENDIX B. ARDUINO CODE FOR IR EMITTER 53

APPENDIX C. PROCESSING CODE FOR LEAP MOTION TEST 59

APPENDIX D. PROCESSING CODE FOR LEAP MOTION DATA COLLECTION 61

APPENDIX E. PROCESSING CODE FOR LEAP MOTION HELO CONTROL 63

APPENDIX F. ARDUINO CODE FOR DIGITAL POTENTIOMETER 69

APPENDIX G. PROCESSING CODE FOR COLOR TRACKING 73

APPENDIX H. PROCESSING CODE FOR DRONE CONTROL WITH ONE WEBCAM 75

LIST OF REFERENCES 83

INITIAL DISTRIBUTION LIST 85

LIST OF FIGURES

Figure 1. Aerial Torpedo. Source: [2]. 2

Figure 2. IAI Scout. Source: [2]. 2

Figure 3. Predator B. Source: [2]. 2

Figure 4. Leap Motion Device. Source: [6]....7

Figure 5. Interaction Area of the Leap Motion Sensor. Source: [6]....8

Figure 6. Leap Motion 3D Representation. Source: [6]. 8

Figure 7. Leap Motion Basic Dynamic Gestures. Source: [6]. 9

Figure 8. Leap Motion Static Gestures. Source: [7]. 10

Figure 9. Arduino Uno. Source: [8]. 11

Figure 10. Arduino Mega. Source: [8]. 11

Figure 11. Processing Related Programing Languages. Source: [9]. 12

Figure 12. Drone Control System with All Major Components 13

Figure 13. FQ777 IR Controlled Helicopter. Source: [12]....15

Figure 14. FQ777 Remote Controller....16

Figure 15. FQ777 Remote Controller Circuit Board....17

Figure 16. IR Signal from the FQ777 Full Throttle Command....18

Figure 17. IR Signal Emitter Schematic Diagram....21

Figure 18. QS5010 Super Mini IR RC Helicopter. Source: [15]. 22

Figure 19. Schematic View of Leap Motion Sensor. Source: [17]. 23

Figure 20. Leap Motion Sensor Cartesian Coordinate Reference. Source: [19]. 24

Figure 21. Magnitude of the Leap Motion Sensor Tracking Points during Stable Hand Test 25

Figure 22. Flow Chart of the Processing Program to Control the FQ777 27

Figure 23. Architecture of the System to Control the FQ777 28

Figure 24. Demonstration of the Operation of the FQ777 Using the Leap Motion Sensor 29

Figure 25. Architecture of the System to Control the JJRC H31 31

Figure 26. JJRC H31 Quadrotor. Source: [20]. 31

Figure 27. JJRC H31 Remote Controller. Source: [20]. 33

Figure 28. Front of the Printed Circuit Board of the H31 Remote Controller 33

Figure 29. Back of the Printed Circuit Board of the H31 Remote Controller 34

Figure 30. Digital Potentiometer Schematic Diagram 35

Figure 31. Digital Potentiometer Test 36

Figure 32. Architecture of the System to Control the JJRC H31 Modified with an Inner Control Loop 38

Figure 33. Test of the Tracking Algorithm Based on Movement 39

Figure 34. Test of the Algorithm Based on Color 40

Figure 35. Flow Chart of the Processing Program to Control the H31 with One Webcam 41

Figure 36. PID Gains Test in X and Y Axes 43

Figure 37. Commands with Respect to the Position of the Hand 47

Figure 38. Commands with Respect to the Orientation of the Hand. Source: [7]. 47

LIST OF TABLES

Table 1. Characteristics of the FQ777 IR-Controlled Helicopter. Source: [12]. 15

Table 2. Example Capture of IR Command.....19

Table 3. Matrix of IR Commands for the FQ777 21

Table 4. Test Results for the Accuracy of the Leap Motion Sensor 24

Table 5. Characteristics of the JJRC H31 Quadrotor. Source: [20]. 32

Table 6. Ziegler–Nichols Method. Source: [23]. 42

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2D	two-dimensional space
3D	three-dimensional space
API	Application Program Interface
CAN	Controller Area Network
FFT	Fast Fourier Transformation
GPS	Global Positioning System
IDE	Integrated Development Environment
I2C	Inter-Integrated Circuit
I/O	Input/Output
IR	infrared
LED	light emitter diode
PID	proportional, integral, and derivative
RF	radio frequency
RC	remote control
SPI	Serial Peripheral Interface
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Xiaoping Yun, and my second reader, Dr. James Calusdian, for their patience and guidance. This thesis research would not have been possible without their help.

I would like to thank the Peruvian Navy for giving me the opportunity to study this master's program. The knowledge learned here will be of great advantage to the Peruvian Navy.

Lastly, I would like to thank all of my family and friends for their support in these two years.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. HISTORY AND APPLICATIONS OF UNMANNED AERIAL VEHICLES

An unmanned aerial vehicle (UAV), hereafter referred to simply as a drone, is a vehicle that does not have a human pilot on board and is capable of being controlled remotely [1]. Drones can autonomously perform complex tasks, such as takeoff and landing, flight stabilization, and point-to-point navigation. They use embedded systems incorporating a variety of onboard sensors, such as inertial sensors, cameras, and GPS [2].

According to Jha in [3], drones first appeared in the mid-1800s. Austria used an unmanned balloon loaded with bombs to attack Venice, Italy [3]. The exact origin of the use of drones is difficult to determine, although WWI saw the first real introduction of an unmanned aircraft in the military service. Toward the end of WWI, the U.S. Army was working on what it called the Aerial Torpedo, shown in Figure 1, a pilotless, small, biplane bomber that essentially worked as a kamikaze drone. The British Army's Aerial Target, introduced at the same time, was a remotely-controlled, unmanned monoplane that demonstrated the use of radio signals to direct an aerial bomb [2]. During the initial years of WWII, the OQ-2, which is the military mission designation for an observation unmanned aerial vehicle, was the first remote controlled aircraft manufactured in the United States by the Radioplane Company. The follow-on model produced by this company was the OQ-3; it was the most prevalent pilotless target aircraft found in U.S. service during WWII [2]. In 1973, after the Yom Kippur war, the Israel Aircraft Industries (IAI) developed the Scout drone, shown in Figure 2. Its two principal characteristics were a very low radar signature and a compact size, which resulted in a challenging target to engage in combat, as well as in targeting practice. In the 1990s, with the availability of GPS, drones were freed from their dependency on inaccurate onboard navigation systems, which were generally based on computerized dead-reckoning [2]. This allowed Abraham Karem to develop the General Atomics' GNAT-750, known to be the predecessor of the Predator, shown in Figure 3, which was capable of aerial reconnaissance and tactical strikes. By the year 2001, the Predator was the most

common remotely-piloted aircraft employed by the United States Air Force and the Central Intelligence Agency in support of military operations in Afghanistan against Al-Qaeda [4].

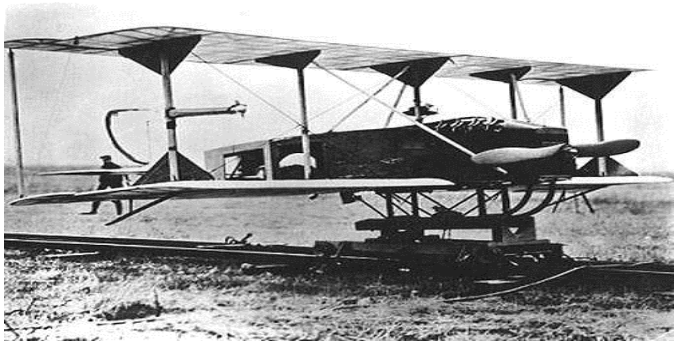

Figure 1. Aerial Torpedo. Source: [2].

Figure 2. IAI Scout. Source: [2].

Figure 3. Predator B. Source: [2].

Over the course of time, as technology evolved, drone capabilities were extended mainly due to the developments in electronics, guidance and navigation, communications, and control [5]. Among the numerous military applications where drones are currently used, the most basic functions include intelligence, reconnaissance, and surveillance; however, military drones are also designed to accomplish combat-related missions, such as target tracking and deployment of defensive and offensive weapon systems [3].

Drone designs and applications are many; therefore, they can be classified in numerous different ways. Most of the literature is in agreement that drones can be classified according to their range of action: high altitude, long endurance (HALE); medium altitude, long endurance (MALE); tactical, close-range; and mini-, micro-, and nano-scale drones. Drones are also classified by their configuration: fixed wing, flapping-wings, blimps, and rotary wing [2].

Today, drones are widely used in many different civilian applications for solving problems and overcoming challenges across numerous industries [1], such as:

- Remote sensing (electromagnetic spectrum analyzers, gamma ray sensors),
- Oil, gas, and mineral exploration,
- Domestic surveillance,
- Policing activities by law enforcement agencies,
- Forest fire detection,
- Precision search and rescue missions,
- Security of pipelines, power lines, coastline and borders monitoring for illegal immigration and imports,
- Delivery systems,
- Scientific research in atmospheric environments.

B. PROBLEM STATEMENT

Military drones in use today have been designed to accomplish a wide array of critical missions. From an engineering standpoint, they are very complex systems that incorporate technologies from a wide range of specialized disciplines, including aerodynamics, structures, propulsion, avionics, and sensors, to name just a few examples. To operate these complex systems requires advanced user interfaces, which may consist of multi-function displays, audio systems, as well as hand- and foot-activated controls. Examples of these are switches, buttons, knobs, throttle, joystick, and rudder pedals. Operators of today's complex drone systems are highly skilled and have received extensive and dedicated training to learn how to pilot modern drones. With this in mind, it is a worthwhile endeavor to explore alternative ways to simplify the user interface with the aim of reducing the degree of skill and training required to operate today's drones. New interface technologies, which are always being released to the market, should be evaluated to determine if they could be used effectively to operate a drone.

C. THESIS GOALS

The main goal of this thesis research is to design a simple and intuitive user interface to pilot advanced drone systems. A main avenue of research will be the integration of a device known as the Leap Motion sensor. The sensor sits on the desktop and monitors a user's hand(s) to provide relative position and orientation information. The measured data of one's hands can potentially be used to direct the motion of a connected drone.

This investigation has three subsidiary goals, as identified below:

1. Establish an interface between the Leap Motion sensor and the computer that allows starting the process of transmitting control commands to the drone.

2. Design and test an algorithm to process the data acquired by the motion sensor.

3. Take over or replace the remote controller of a radio frequency/infrared (RF/IR) controlled drone by means of modifying its internal circuitry or synthesizing communication protocols.

D. ORGANIZATION

This thesis is divided into four chapters. In addition to this introduction chapter, a description of the Leap Motion sensor, the Arduino microcontroller, and Processing software programing language is provided in Chapter II. The system design and integration of the Leap Motion sensor with a desktop computer is discussed in Chapter III. Two different approaches used to take over an IR-controlled helicopter drone and a RF-controlled quadrotor drone are also discussed in Chapter III. A summary and a discussion of future research derived from this project are provided in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

II. INPUT DEVICE AND SOFTWARE

The hardware and software selected to accomplish the stated thesis research goals are presented in this chapter. For the hardware, the Leap Motion sensor was utilized as it provided a user-friendly interface that could be used for drone motion control. Additional hardware utilized were microcontrollers from the Arduino series. These devices are low-cost, easy to integrate with external hardware, and come with a wide range of documentation and example applications. To integrate the hardware, an open-source programming language known as Processing was used. This software programming language was selected because of its similarities with the Arduino software language.

A. LEAP MOTION SENSOR

The Leap Motion sensor, shown in Figure 4, consists of three infrared light-emitting diodes (LED) and two infrared cameras. The cameras track reflected infrared light at an operating wavelength of 850 nm, which is outside the visible light spectrum [6]. The sensor is used to report precise position coordinates of one's hands located within its field-of-view above the sensor.

Figure 4. Leap Motion Device. Source: [6].

Because of its wide-angle lenses, the interaction area, which takes the shape of an inverted pyramid, is approximately 2×2×2 cubic feet, as shown in Figure 5.

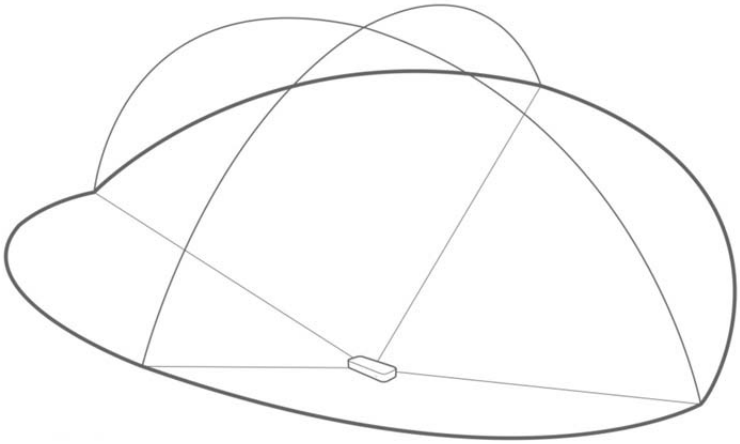

Figure 5. Interaction Area of the Leap Motion Sensor. Source: [6].

The device sends the acquired sensor data to a connected computer via USB, where it is then analyzed using digital image processing algorithms to compensate for background objects (such as a person's head) and ambient light. The final product of this process is a three-dimensional (3D) representation of what the device sees, as shown in Figure 6.

Figure 6. Leap Motion 3D Representation. Source: [6].

The Leap Motion Application Program Interface (API) provides a collection of protocols and routines for developing user applications. The building blocks of the Leap

Motion API are three different classes: arms, hands, and fingers. At the same time, each of these classes is divided into subclasses with different attributes and properties that provide high precision information about position, orientation, and movement of the user's hands.

One of the more advanced features provided by the Leap Motion API is dynamic gesture recognition. This is achieved by analyzing over time the movement of individual fingers, hand rotation, and arm orientation [6]. The Leap Motion API identifies four basic gestures as shown in Figure 7.

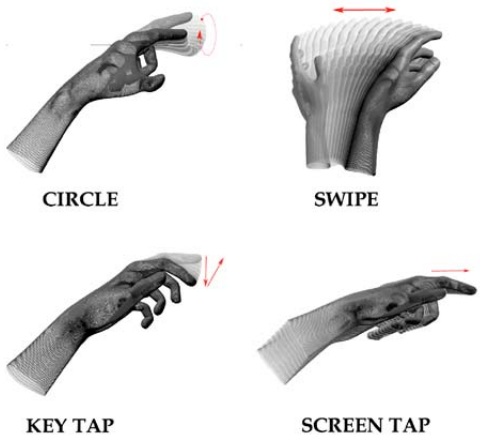

Figure 7. Leap Motion Basic Dynamic Gestures. Source: [6].

Furthermore, in [7], static gestures are proposed to be utilized by the human operator to perform different tasks. Static gesture recognition is developed based on relative distance between one's fingers and the center of the hand. Examples of static gestures are shown in Figure 8.

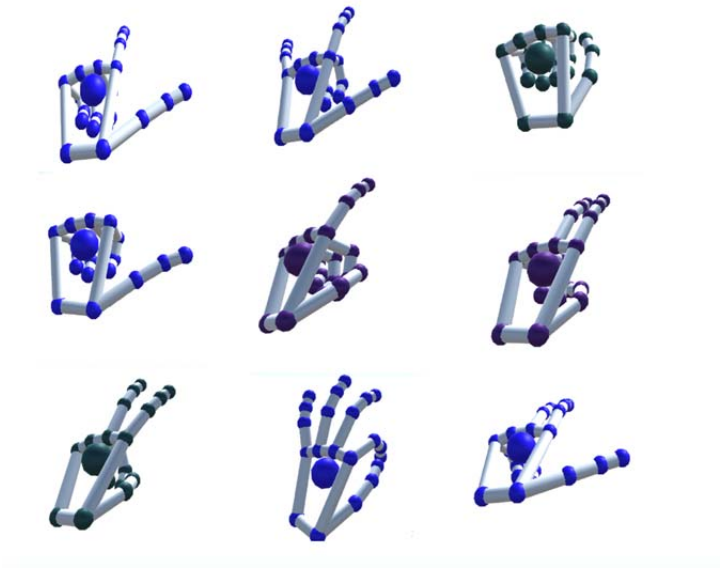

Figure 8. Leap Motion Static Gestures. Source: [7].

B. ARDUINO MICROCONTROLLER

An Arduino microcontroller is a circuit board that can be programmed to do many different things. It can read information from sensors, like motion sensors, photo sensors, pressure sensors, or GPS receivers. Outputs are also available to control devices, like motors and servos. A number of popular communication protocols are also available on the microcontroller, such as serial, inter-integrated circuit (I2C), controller area network (CAN), and serial peripheral interface (SPI). The Input/Output (I/O) characteristics of the Arduino-series of microcontroller boards make it easy for anyone to connect the physical world around us to the digital world. In summary, an Arduino microcontroller is an open-source development platform. It integrates a software and hardware interface that is easy to program [8].

For this thesis work, we use two models of Arduino boards, but there are many different models available in the market. The major differences between them are built-in features like Wi-Fi, Bluetooth, Ethernet, number of I/O ports, and amount of memory.

The boards used in this thesis research are the Arduino Uno, shown in Figure 9, which is the most common microcontroller board on the market. The other is the Arduino Mega, shown in Figure 10. This one is used mainly for more advanced projects. The

Arduino Mega has a more powerful processor and provides additional I/O pins, which are features not found on the less capable Arduino Uno board.

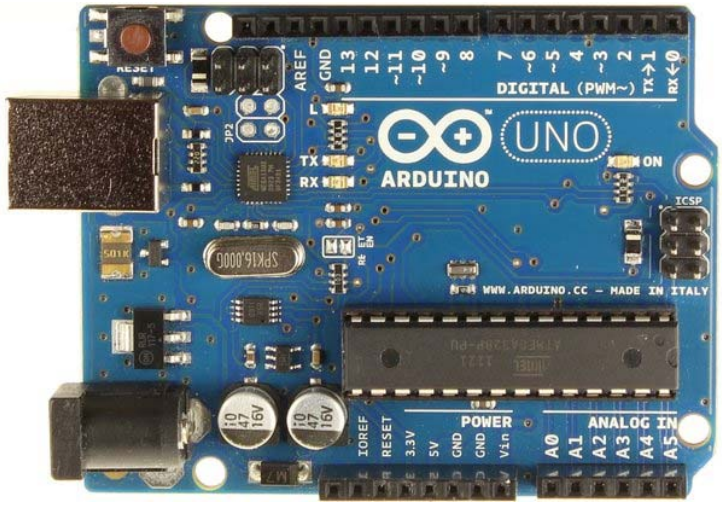

Figure 9. Arduino Uno. Source: [8].

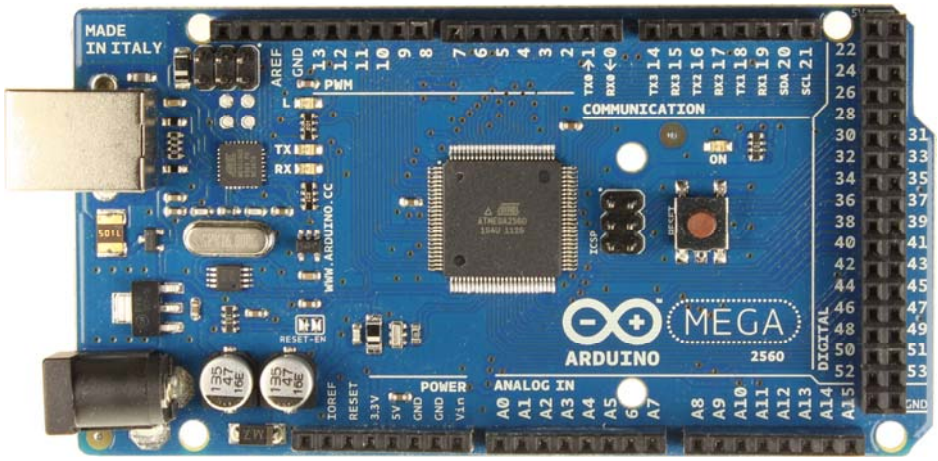

Figure 10. Arduino Mega. Source: [8].

C. PROCESSING SOFTWARE

Processing is an open-source software application used by developers to write, edit, compile, and execute Java code. This software was developed by Reas and Fry in the MIT Media Lab in 2001 [9].

The Processing functions provide the tools for the elaboration of on-screen graphics, which enables immediate visual feedback of what the code is executing in real time. Processing employs the same principles, structures, and concepts as many other programing languages, like Java for example.

Although its syntax is based on Java, Processing adds custom features related to graphics and collaboration [10]. From Figure 11, we see that Processing has a large family of related languages and programming environments. In the decade of the 1980s, one of the most important programming languages that influenced Java, and ultimately Processing, was the programming language C; however, in comparison with C, Processing greatly reduces the number of lines of code needed to implement a program. In 2003, Barragan [11] initiated a project called Wiring, which was based on the Processing language and was destined to become a programing framework for different kinds of microcontrollers. The Arduino project, based on Wiring, uses the Processing Integrated Development Environment (IDE) with a primitive version of the C-programing language.

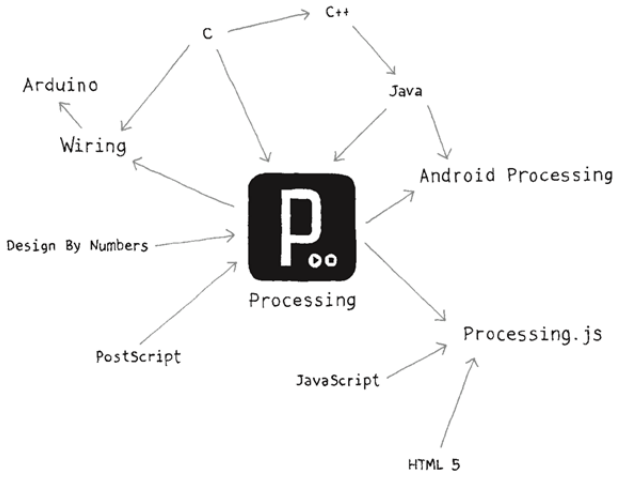

Figure 11. Processing Related Programing Languages. Source: [9].

III. SYSTEM DESIGN AND IMPLEMENTATION

To interface the Leap Motion sensor with a computer in order to control a drone, it was necessary to carry out the design, development, and testing of several systems that utilize both hardware and software components. For this thesis research, an Arduino-embedded microcontroller was programmed to process commands it received from a desktop computer and to generate the commands required to control the motion of a drone. First, prototype electronic circuits using the microcontroller and other peripheral hardware were constructed and tested. Second, in order to track the position and orientation of the drone, an optical tracking system was developed from a low-cost webcam, which was readily available in the laboratory. This aspect of the design required testing of available image processing algorithms to find the best approach to track the drone within the webcam's field-of-view. Third, the Leap Motion sensor was integrated into the design using a software program developed for this thesis research. The system diagram, shown in Figure 12, presents all of the major components developed and integrated for the demonstration of the drone controller.

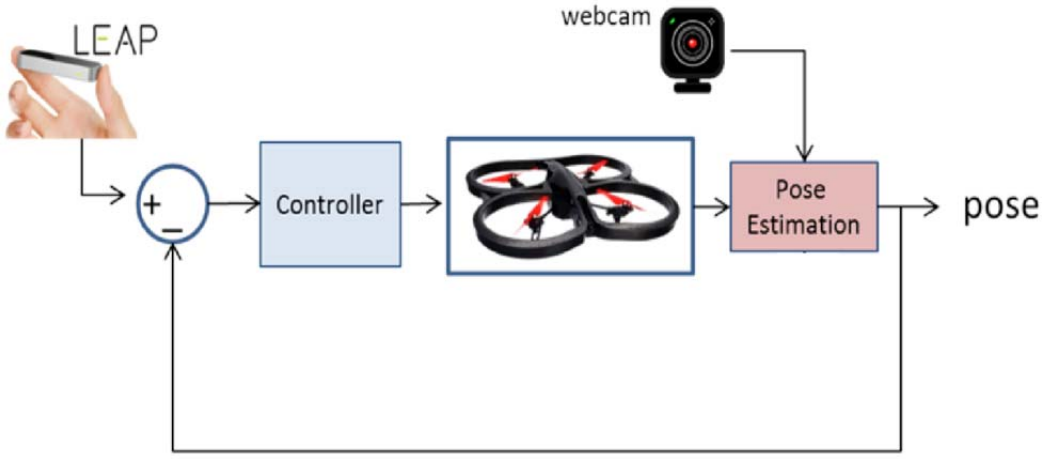

Figure 12. Drone Control System with All Major Components

This chapter is divided as follows. The synthesis of the command signals required to operate an IR-controlled drone helicopter is explored in Section A. Several prototype test circuits were constructed to 1) capture and store the control signals for the IR-controlled drone, and 2) synthesize the required command signals using embedded hardware and software. Also described in this section is the manner in which the Leap Motion sensor was integrated into the design using the open-source programming language called Processing. In Section B, an improved open-loop control system is proposed using a quadrotor to increase precision in control. In Section C, the description and testing of a closed-loop control system based on one webcam is discussed. This alternative design employed one stationary camera to track and maintain the orientation of a drone at a fixed location. The information derived from the fixed webcam provided sensor information to enable automatic control of the drone position and orientation within the camera's field of view.

A. OPEN-LOOP CONTROL OF THE FQ777 INFRARED-CONTROLLED HELICOPTER

1. Introduction

The main objective of this section was to study and develop the prototype hardware and software required to control the motion of a drone. In order to achieve this goal, we first had to understand how the drone was controlled. In other words, we needed to understand the communication protocol between the remote controller and the drone and to come up with a solution to synthesize this type of communication. Due to time, budget, and hardware availability, a small, low-cost, IR-controlled helicopter was utilized for this part of the project.

2. Implementation

In search of a low-cost helicopter that met the basic requirements, which were that it was IR-controlled and have at least three control channels, we decided to use the FQ777 IR control helicopter, shown in Figure 13. Some of its characteristics are presented in Table 1.

Figure 13. FQ777 IR Controlled Helicopter. Source: [12].

Table 1. Characteristics of the FQ777 IR-Controlled Helicopter. Source: [12].

Brand Name	SBEGO
Type	Helicopter
Model	FQ777-610
Controller Mode	MODE 2
State of Assembly	Ready-to-fly (RTF)
Fly Time	8 min
Charging Voltage	3.7 V
Dimensions	9.05 × 5.71 × 3.9 inches
Motor Type	Brush Motors
Material	Plastic, Metal
Power Source	Electric
Remote Distance	10 - 12 meters
Control Channels	3 Channels
Feature	6-axis gyro control system

From Figure 13, we see that the FQ777 has coaxial rotors; this means a pair of helicopter rotors mounted one on top of the other on concentric independent and counter-rotating shafts. This type of configuration provides balance of torques around the central axis, reduction of noise, and also prevents the aerodynamic phenomenon called Dissymmetry of Lift that affects helicopters with simple rotor configurations [13].

From Table 1, we observe that the FQ777 is a three-channel coaxial helicopter, which means that there are three channels on the remote controller, shown in Figure 14, to control the motion of the helicopter. On this type of controller, channel one controls the throttle (up/down), channel two controls the yaw (left/right), and channel three controls the pitch (forward/backward). A small horizontal propeller at the tail that rotates clockwise or counter-clockwise controls the forward and backward movement, respectively. From the flying tests performed, we observed that the hovering was extremely stable, but forward and backward speeds were limited.

Figure 14. FQ777 Remote Controller

To investigate the remote controller in more detail, we opened it and found that it had three infrared light emitter diodes (IR LED) mounted at the top of the circuit board, as shown in Figure 15. These LEDs were used to transmit IR signals to the helicopter. In general, better performance is achieved when a greater number and quality of IR emitters is utilized in the transmitter. Apart from the number of IR LEDs, we were not able to get more useful information from this preliminary and simple investigation.

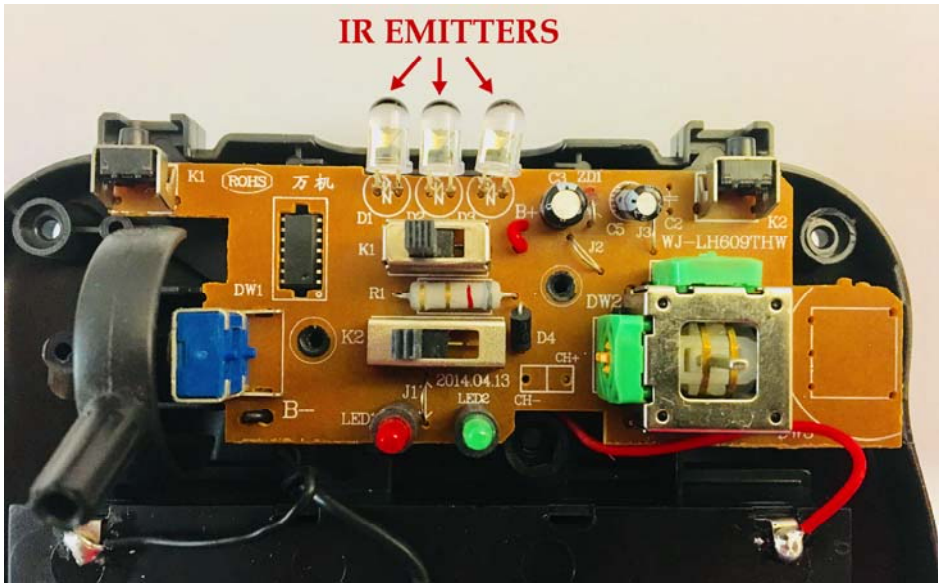

Figure 15. FQ777 Remote Controller Circuit Board

The most common consumer IR device used in our daily life is the television remote controller, which uses the infrared electromagnetic spectrum to wirelessly communicate instructions to the television. Over time, different television manufacturers have developed different communication standards; this is evidenced from the fact that a remote controller from a Sony television cannot be used to control a Panasonic television because they use different transmission protocols. This information led us to assume that it was more likely that the different remote-controlled drone manufacturers also use their own proprietary IR transmission protocols. Later in the testing and development, this assumption was proved to be correct.

At this point, we assumed that there were a variety of IR transmission protocols used by each manufacturer, and we had not yet identified the one used by the FQ777. We needed to identify a way to be able to send the IR commands from the computer to control the helicopter. For this matter, two different options were explored:

1. Use a signal analyzer or oscilloscope to visually evaluate the binary pulses to determine the transmission protocol, or
2. Use an IR signal capture device or circuit to capture the IR signals from the remote controller and store them in a file on a computer or Arduino.

While the first option seemed to be the most logical path to follow, it turned out to be the more challenging to achieve because of the large variety of parameters present in an IR transmission protocol. It was necessary to visually resolve the encoding standard: pulse coded, space coded, or shift coded. Then the start pulse signal needed to be located. It was also necessary to determine the number of bits used to transmit a single signal frame: 8, 12, 16, 20, 24, or 32. Then it was required to define the time used to represent a logical one and zero, respectively. Finally, we needed to determine the time between bits. As an example of this process, an oscilloscope image of the IR signal frame corresponding to the full throttle command of the FQ777 remote controller is shown in Figure 16. A complex bit pattern would have to be captured and decoded to reverse-engineer the protocol, which made this approach very difficult. The second option was considered to be the better choice and much simpler to apply.

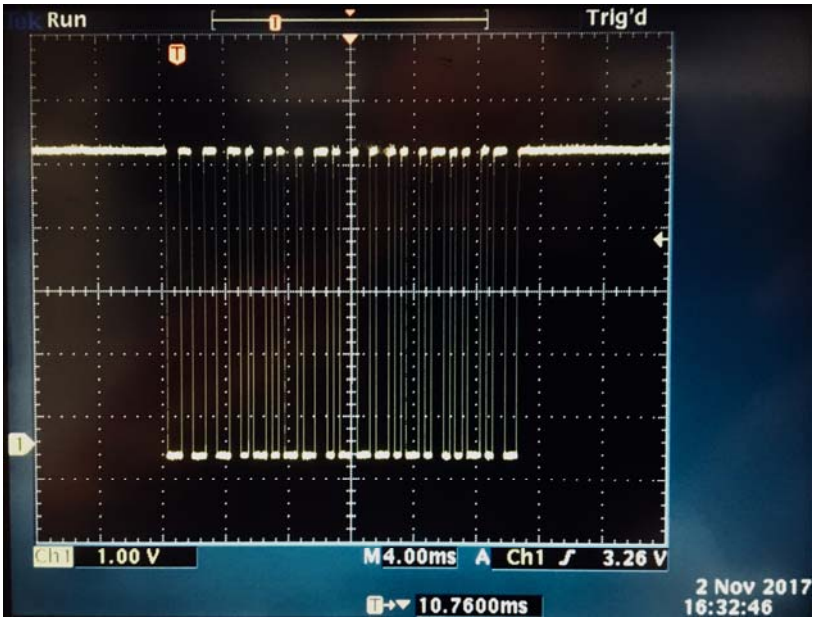

Figure 16. IR Signal from the FQ777 Full Throttle Command

a. IR Signal Capture

To employ the second option, we needed a device or circuit to capture IR signals. For this purpose, a simple circuit suggested by Angel [14] was built. It was interfaced to

an Arduino Uno. The operation of the signal capture circuit was tested by sending different IR commands with the FQ777 remote controller to be captured and stored in the memory of the microcontroller. The captured signal was then retransmitted by the Arduino microcontroller to verify if the helicopter was able to decode the raw data. This test was successful with all of the IR commands that were examined. An example of a typical IR command code that was captured with the IR signal capture circuit is shown in Table 2. This sequence of numbers represents the byte values recorded by the test circuit when the left joystick was in the down position and the right joystick was in the center position.

Table 2. Example Capture of IR Command

800 750 750 800 700 850 300 500 650 500 300 450 650 550 250 500 300 500 250 500 300 500 650 500 300 500 650 500 300 850 300 500 250 900 650 500 650 900 250

b. IR Command Code Matrix and IR Signal Emitter

The limitation of the Arduino Uno microcontroller used in the IR signal capture circuit is that it can only store one IR command in memory at a time. Rather, we needed to store a sufficient number of codes in memory to be able to fly the helicopter with sufficient precision. To solve this problem, we created a matrix of codes with each IR command represented by a single character, as shown in Table 3. The elements of the [10×5] command matrix represent a set of 50 command codes that are used to transmit one of 50 possible corresponding IR commands. This table takes into consideration a sufficient level of precision needed to fully control the helicopter and the maximum availability of memory in the microcontroller. The corresponding IR commands for each element in the command matrix in Table 3 are found in greater detail in Appendix A.

The matrix of IR commands, presented in Table 3, was designed by dividing the throttle potentiometer of the FQ777 remote controller into ten different levels, with one as the lowest and ten the highest. For each of these ten throttle levels, there were five different possible IR commands:

1. Yaw and pitch in the center position, which caused the helicopter to hover at a desired height,
2. Yaw turning right and pitch in the center position, meaning that the helicopter turned to the right while maintaining altitude,
3. Yaw turning left and pitch in the center position, meaning that the helicopter turned to the left while maintaining altitude,
4. Yaw in the center position and pitch forward, meaning that the helicopter moved forward while maintaining altitude, and
5. Yaw in the center position and pitch back, meaning that the helicopter moved backward while maintaining altitude.

The next step was to create a circuit dedicated exclusively to transmitting the raw IR command code signals stored in the matrix. The IR signal emitter schematic diagram circuit is shown in Figure 17.

For the circuit shown in Figure 17, we initially intended to use the Arduino Uno, but because of serious memory limitations, as well as reduced data processing speed, we decided to use the Arduino Mega to obtain more favorable results.

The Arduino Mega was programed so that when it received a valid command code character corresponding to Table 3 via the serial port, it searched for the corresponding raw data stored in a list of arrays representing the matrix of IR commands previously created from the FQ777 remote controller. The microcontroller then transmitted the corresponding IR command via the IR emitters. The Arduino code for the IR signal emitter circuit is found in Appendix B.

Table 3. Matrix of IR Commands for the FQ777

	THROTTLE	RIGHT	LEFT	FWD	BACK
1	Q	A	Z	e	o
2	W	S	X	f	p
3	E	D	C	g	q
4	R	F	V	h	r
5	T	G	B	i	s
6	Y	H	N	j	t
7	U	J	M	k	u
8	I	K	b	l	v
9	O	L	c	m	w
10	P	a	d	n	x

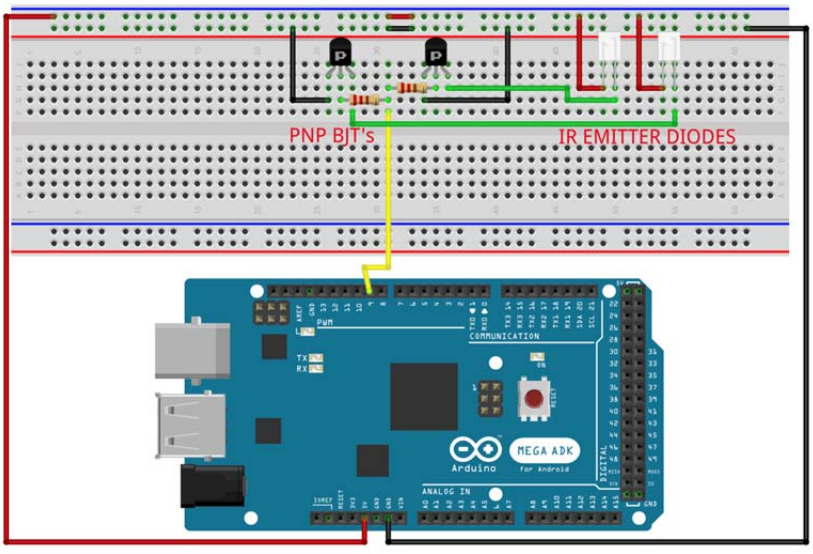

Figure 17. IR Signal Emitter Schematic Diagram

At this point we were able to confirm our initial assumption that different remote-controlled drone manufacturers used their own proprietary IR transmission protocols. For this matter, we tried to operate the QS5010 Super Mini IR remote controlled helicopter, shown in Figure 18, because it shared all of the basic common characteristics with the FQ777. Testing the operation of the IR signal emitter circuit with the QS5010 was not successful.

Figure 18. QS5010 Super Mini IR RC Helicopter. Source: [15].

c. Leap Motion Sensor Integration

In this section, we describe the next step in the development, which was to integrate the Leap Motion sensor. To do this, we used a programming language called Processing because of its similarities with Arduino. In fact, the appearance of the user interface for the IDE for Processing and Arduino are nearly identical.

In order to interface the Leap Motion sensor with the computer, we incorporated a software library called Leap Motion for Processing, designed by Morawiec [16]. This library allowed us to receive detailed information, such as position, orientation, and movement of the fingertips, center of hands, arms, etc. This information was received in the Cartesian coordinates relative to the center point (center IR LED) of the Leap Motion sensor's field-of-view [17], as shown in Figure 19.

We took into consideration that the Leap Motion sensor was a gesture-based interface with sub-millimeter accuracy of 0.7 mm [17], and it very accurately tracked the motion of the human hand. Furthermore, according to [18], the human hand exhibits a

natural tremor behavior, which is the involuntary movement of muscles. From the study, the tremor amplitude was reported to vary between 0.4 mm ± 0.2 mm for children and 1.1 mm ± 0.6 mm for adults. To avoid the tremor of the hand to transfer into a jittering movement of the helicopter, various techniques have been proposed. As an example, Hayden [19] proposed to use the Fast Fourier Transformation (FFT) with a band pass filter of 0.5 Hz to filter and eliminate the noisy measurement obtained from the Leap Motion sensor to control the six degrees-of-freedom (DOF) Jaco Arm manipulator.

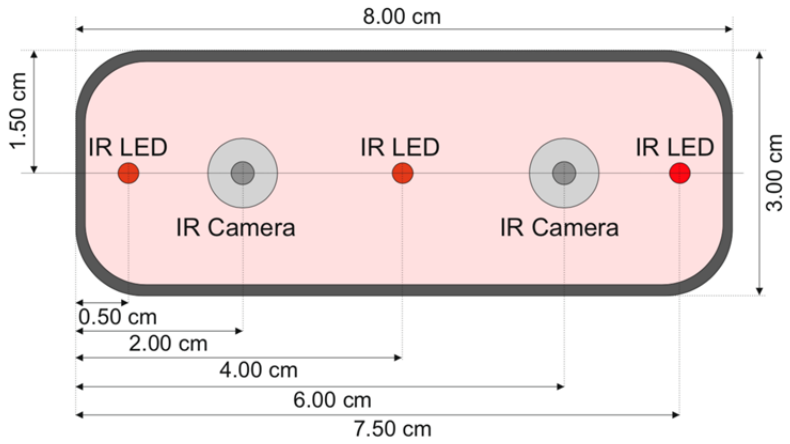

Figure 19. Schematic View of Leap Motion Sensor. Source: [17].

The first program we wrote for the Leap Motion sensor was designed to check the accuracy of the data acquired and observe if the sensor was able to detect the tremor of the human hand. To verify this, we performed an experiment to collect 500 samples in the X, Y, and Z axes while holding the right hand in a constant position relative to the sensor. The sensor coordinate system in which the measurements were made is shown in Figure 20. The results of the experiment are summarized in Table 4, and the Processing code used for this experiment is found in Appendix C.

Table 4. Test Results for the Accuracy of the Leap Motion Sensor

	X-axis (cm)	Y-axis (cm)	Z-axis (cm)
Median	52.03734400	63.89045550	43.52832200
Stand. Dev.	0.13955226	0.22426161	0.26207200
Max	52.27108400	64.38776400	44.00585000
Min	51.65746000	63.42564400	43.07614500

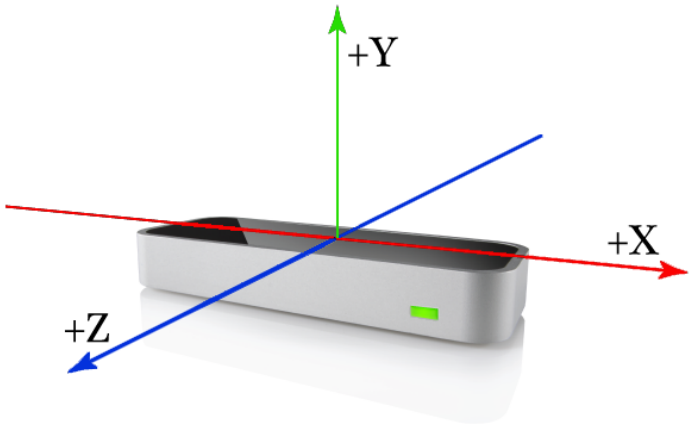

Figure 20. Leap Motion Sensor Cartesian Coordinate Reference. Source: [19].

A plot of the data collected during this experiment is shown in Figure 21. We see that while holding the hand firmly in a constant position with respect to the sensor, slight variations in the X, Y, and Z distance were measured.

To facilitate integration with the helicopter, an exclusion (or dead) zone was defined. When the hand was detected within this zone, the controller program did not command any motion of the drone. This provided a zone by which the operator could quickly stop the motion of the helicopter if required. It also created an entry point from where to begin operation of the helicopter. The exclusion zone also ensured that the helicopter did not respond to data from this region where the sensor was determined to be less accurate [19]. The exclusion zone had dimensions 10×10×10 cm from the origin of the sensor reference.

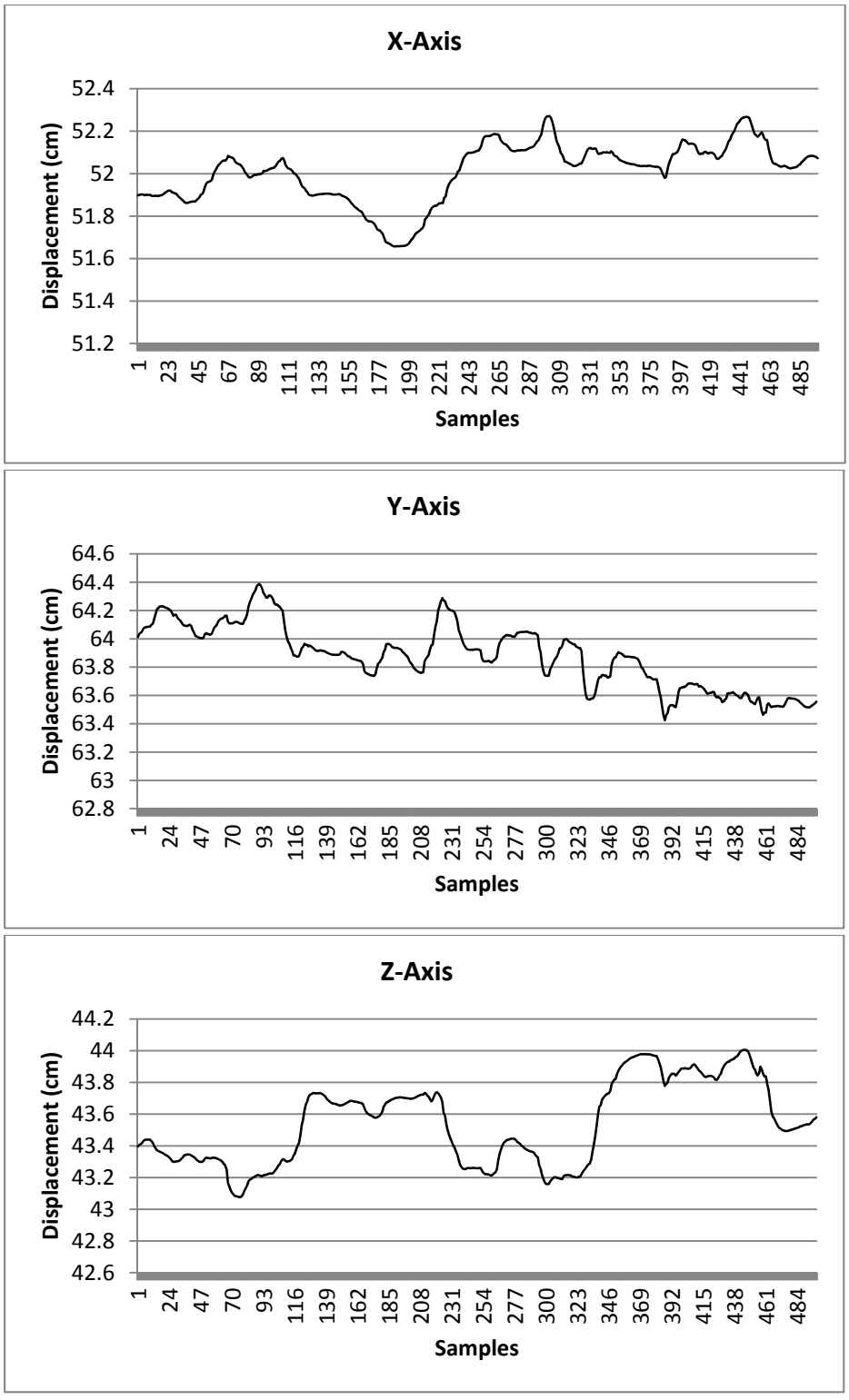

Figure 21. Magnitude of the Leap Motion Sensor Tracking Points during Stable Hand Test

Next, it was necessary to quantize the data obtained from the sensor and map the Y-axis distance to the throttle levels of the command matrix in Table 3. To obtain the desired quantized result, we used a built-in Processing function called *map( )* that remapped a number from one range to another. The data type of the output number from this function was of type float. For this reason and to avoid jitter in the output, we rounded the result by converting it into an integer. As an example, when the output value from the *map( )* function was 4.43, it was automatically rounded to 4, matching precisely the fourth throttle level from the matrix of IR commands for the FQ777. The minimum and maximum output values of the function were set to zero and ten, respectively. The Processing code used for this part of the experiment is found in Appendix D.

d. Open-loop System Integration

The work up to this point has been to design and construct the basic elements required for the open-loop operation of the IR helicopter. This required construction, programming, and testing of the IR emitter signal prototype circuit using the Arduino Mega microcontroller. The other major component developed was the Processing language computer program designed to interface the PC with the Leap Motion sensor. In the next phase of the development effort, we worked to integrate these basic elements into a working prototype demonstration of the drone controller interface.

In this part of the development, a Processing language program was written to encode the hand position above the Leap Motion sensor into a character command from Table 3 and transmit it to the Arduino Mega microcontroller. The command characters were transmitted via serial communication from the desktop PC to the Arduino microcontroller. When a valid command character was received by the microcontroller, it was translated into the corresponding IR command and transmitted to the IR helicopter. A flow chart of the program that was developed to operate the IR helicopter with the Leap Motion sensor is shown in Figure 22. The final Processing language code for controlling the helicopter is found in Appendix E.

The architecture of the open-loop system is shown in Figure 23. It includes all of the hardware (Leap Motion sensor, desktop PC, and Arduino board with peripheral

hardware) and software (Processing and Arduino code) as well as the communication protocols used between them. The final result is an open-loop system designed to operate the IR helicopter using the Leap Motion sensor.

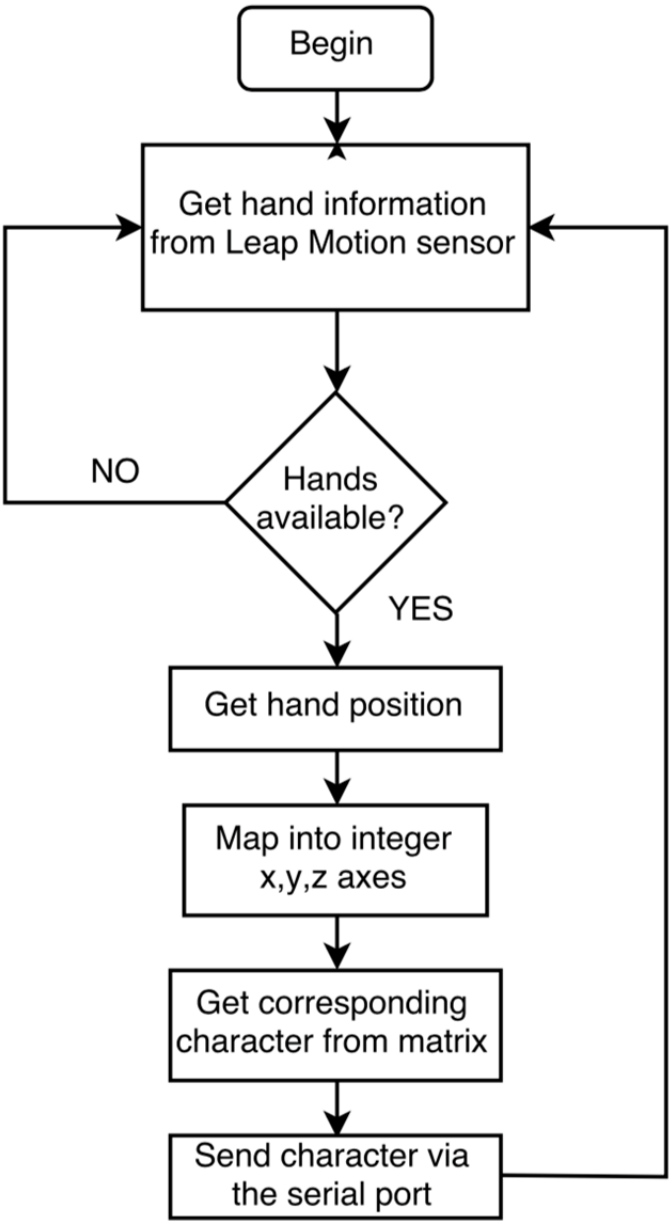

Figure 22. Flow Chart of the Processing Program to Control the FQ777

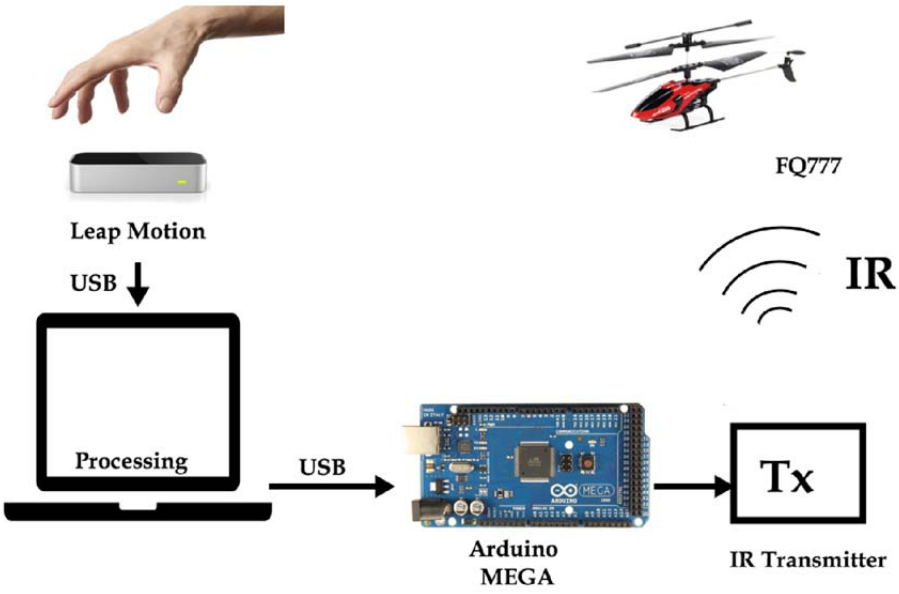

Figure 23. Architecture of the System to Control the FQ777

3. System Testing

To test the final system design, we first tested each module separately and then tested the system as a whole. We began by testing the stability of the interface between the Leap Motion sensor and the computer by capturing many data samples at different positions of the hand above the sensor. The interface proved to be reliable throughout the preliminary analysis of the acquired data. From the Leap Motion tests performed in [19], hand positions less than 11.0 cm above the sensor were reported to be less accurate. Our observations showed similar results, and we avoided this range of measurements by implementing the exclusion zone as described earlier.

The next step was to test the interface between the computer and the Arduino microcontroller. This interface was established via a Universal Serial Bus (USB) with the data transmission rate set to 115,200 bits per second (baud). The test performed here was designed to repeatedly send a predefined array of characters from the PC to the Arduino via the serial port and then to verify if they were received in the same order in which they were transmitted. The results of this test were alarming because the sequence of characters received by the Arduino was completely random. These results made us

realize that we were causing an overflow in the Arduino serial port buffer. This overflow occurred because we were writing and sending data much faster than the Arduino was able to process. To solve this problem, we modified the code to introduce a 70.0 ms delay in the main programming loop. We repeated the test and verified that the buffer overflow had been resolved.

The last step was to verify the operation of the open-loop system with all of its parts together. The prototype system was demonstrated to work as designed. During this test, we were able to continuously control the FQ777 helicopter for approximately seven minutes while controlling the position and orientation of the helicopter as desired. A demonstration of the operation of the IR helicopter using the Leap Motion sensor is shown in Figure 24.

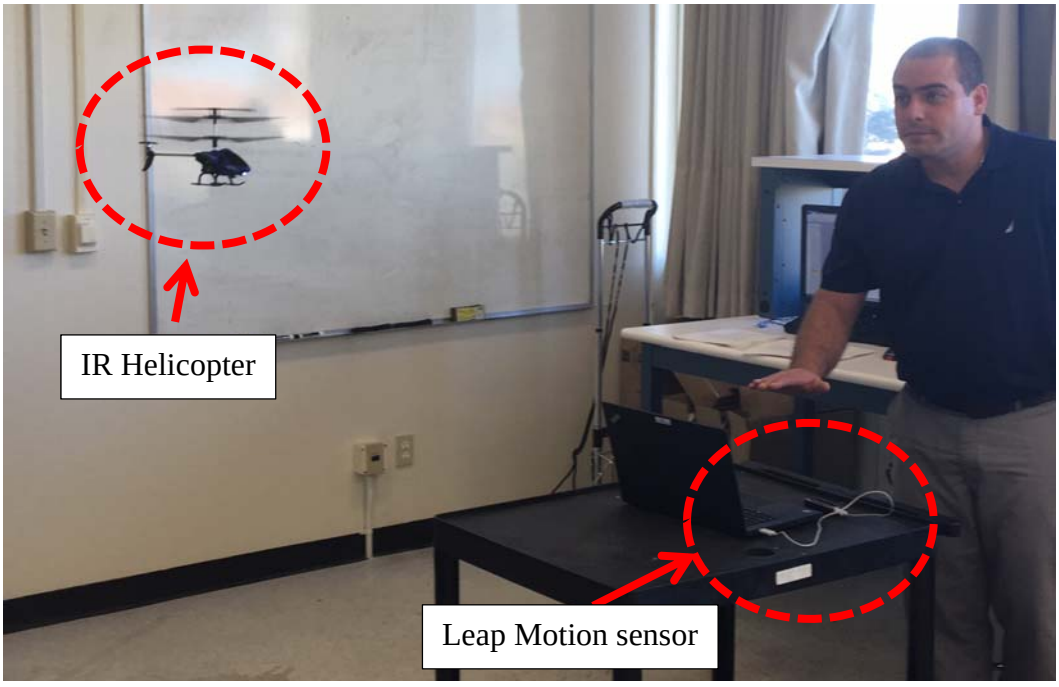

Figure 24. Demonstration of the Operation of the FQ777 Using the Leap Motion Sensor

In summary, we found that we were able to control the IR helicopter by means of the Leap Motion sensor. Furthermore, we predict with confidence that the method

described in this section can be applied to different kinds of IR controlled airborne or ground-based vehicles.

The only limitation observed in this approach was the resolution of the programmed command matrix, which depended on the amount of data storage memory available in the microcontroller. The resolution affected the range of possible IR signals that could be obtained from the two potentiometers of the remote controller. In this experiment, we limited the resolution to only ten levels for throttle and one level each for pitch and yaw. In the next section of this chapter, we address this problem by incorporating a digital potentiometer with a resolution of 156 steps for each control channel (throttle, yaw, pitch, and roll).

B. OPEN-LOOP CONTROL OF THE JJRC H31 QUADROTOR

1. Introduction

In the previous section, we demonstrated that it was possible to control an IR helicopter by means of the Leap Motion sensor using an IR signal emitter circuit. In this section, we explore the possibility of increasing the capabilities of the system designed in the preceding section by integrating a quadrotor that was operated through a radio frequency (RF) link. The architecture of the proposed system is shown in Figure 25. This proposed system has two main advantages. First, since the drone was controlled via an RF link, the expected operation range or distance is greater. Second, the quadrotor has four flight control channels (throttle, yaw, pitch, and roll). With this added degree of motion, we expect to find a greater degree of control. To demonstrate this alternative design, we developed a new technique to operate the drone through modification of the drone's remote controller unit without the manipulation of the RF protocol signal.

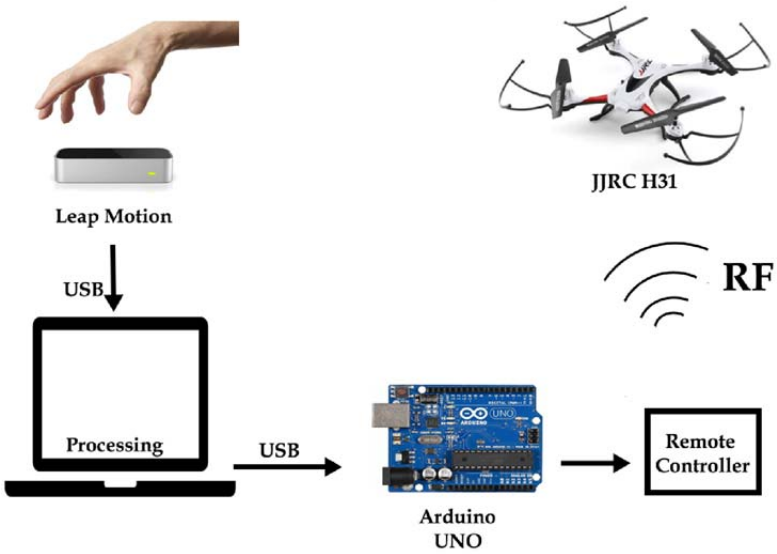

Figure 25. Architecture of the System to Control the JJRC H31

2. Implementation

In Section A, we demonstrated how to control a three-channel drone (throttle, yaw, and pitch), and for this part of the development project we improved the degree of control by incorporating one more control channel (roll) into the system. For this reason, we decided to use the JJRC H31 quadrotor, shown in Figure 26. Some of its major characteristics are presented in Table 5.

Figure 26. JJRC H31 Quadrotor. Source: [20].

Table 5. Characteristics of the JJRC H31 Quadrotor. Source: [20].

Type	Quadrotor
Brand	JJRC
Model	H31
Motor Type	Brushed
Material	Plastic
Frequency	2.4 GHz
Control Channels	Four Channels
Controller Mode	Mode 2
Remote Distance	30 - 40 m
Battery	3.7 V
Fly Time	7 - 8 min
Weight	73 g

From Figure 26, we can see that the H31 quadrotor has an *X* configuration, meaning that there are two frontal motors and two rear motors as opposed to the *cross* configuration that has a pair of motors aligned with the pitch axis while the other two are aligned with the roll axis. The *X* configuration of the H31 makes it more responsive and has better overall performance [2].

The H31 is a four-channel quadrotor. Correspondingly, there are four channels on the remote controller, shown in Figure 27, to control the throttle, yaw, pitch, and roll of the drone. Channel one controls the throttle, channel two controls the yaw, channel three controls the pitch, and channel four controls the roll.

Following the same steps as in the previous section, we decided to open the remote controller and found that the blue antenna extending from the case and the two upper knobs were not attached to the circuit board. We concluded that they were merely a decoration on the remote controller. Inspecting the electronic printed circuit board inside the remote controller, we found two dual-axis potentiometer joysticks, which corresponded to the four channels of control for the H31 quadrotor, as seen in Figure 28. From Figure 29, it can be seen that the actual antenna for the remote controller was found

printed onto the board. Additionally, we noticed that a pad was made on the same path of the printed antenna to allow the possibility for the user to install an external antenna to increase the range and quality of the transmission.

Figure 27. JJRC H31 Remote Controller. Source: [20].

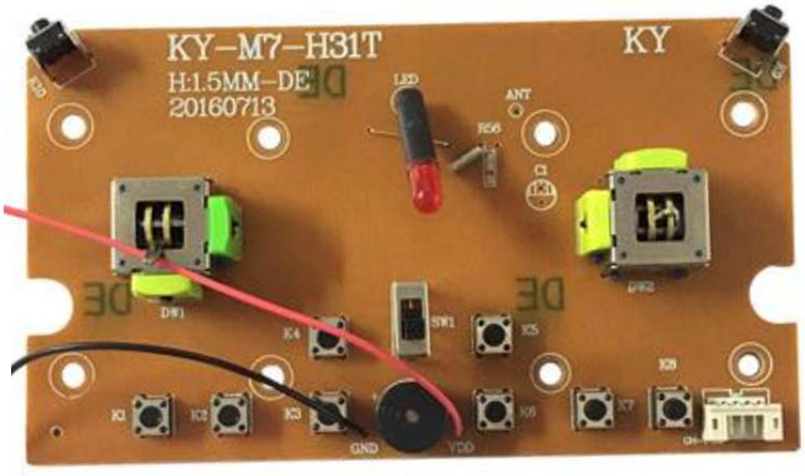

Figure 28. Front of the Printed Circuit Board of the H31 Remote Controller

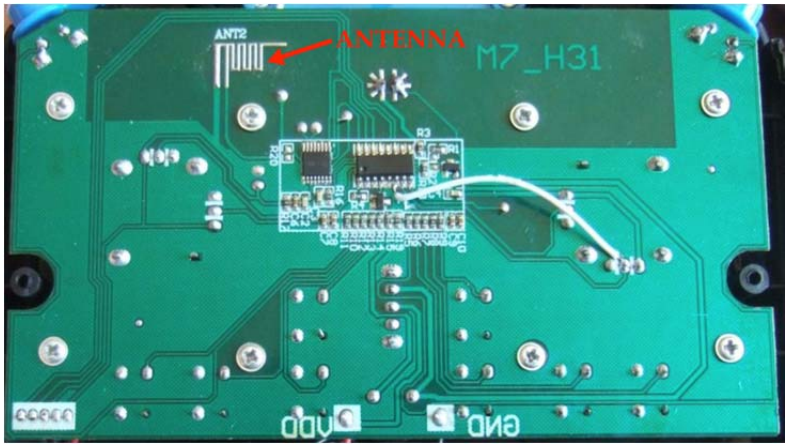

Figure 29. Back of the Printed Circuit Board of the H31 Remote Controller

a. Digital Potentiometer Circuit

Having recognized the limitations in control resolution of the method used in Section A, we proposed a new approach to take over the remote controller. This method involved replacing the two dual-axis analog potentiometer joysticks on the printed circuit board with a 4-channel digital potentiometer that was controlled by an Arduino Uno microcontroller. The digital potentiometer used in this circuit was the AD8403A50 manufactured by Analog Devices [21], shown in Figure 30. This potentiometer produced the same signal as the analog potentiometers on the joysticks but by digital means. It also provided a resolution of 156 different steps in each of the four independent control channels. This was an improvement over the ten steps we used in the previous design.

The Arduino Uno was programed so that when it received an array of commands from the serial port, it checked them for validation and then set the corresponding potentiometer parameters for each of the four channels (throttle, yaw, pitch, and roll) as required. The Arduino code for the digital potentiometer is found in Appendix F.

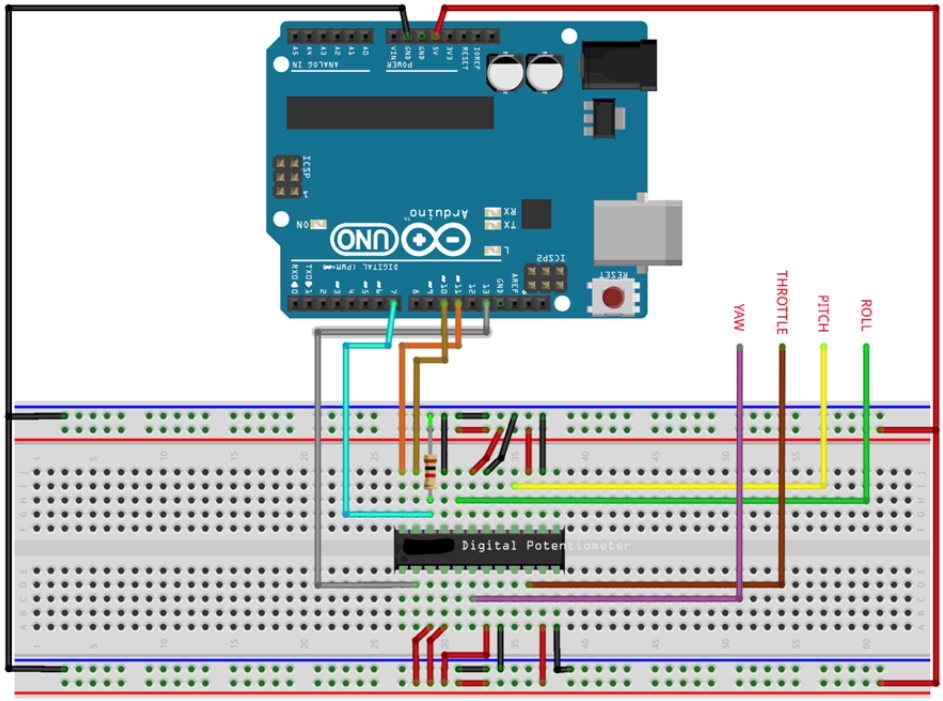

Figure 30. Digital Potentiometer Schematic Diagram

b. Processing and Leap Motion Sensor

To obtain the Leap Motion sensor data, we employed the same process as in the previous section for the FQ777 drone using a custom Processing language program. We began by establishing the exclusion zone of 10×10×10 cm from the origin of the sensor reference in the Processing code. The data collected in each of the three axes was remapped and then rounded to match with the 156 different possible settings for the throttle, roll, and pitch of the digital potentiometer. The information for the yaw channel was obtained from the horizontal orientation of the hand, and this value was also remapped and rounded for the digital potentiometer setting.

The next step was to create an algorithm responsible for compiling the values of the four channels in one array of 12 valid digits. This array was then sent to the Arduino Uno board via the serial port.

3. System Testing

Following the same steps as in the previous section, we first tested each module separately and then tested the system as a whole. We began by testing the interface between the Arduino Uno and the digital potentiometer. A test program was written to transmit a sequence of settings for each of the four command channels of the AD8403A50 digital potentiometer. The transmissions were staged 100 ms apart. The results of the test led us to conclude that the outputs of the digital potentiometer were accurate. Two of the four AD8403A50 outputs were captured on the oscilloscope and are displayed in Figure 31.

Figure 31. Digital Potentiometer Test

The next step was to test the interface between the PC and the Arduino board. This interface was established via USB with the data transmission rate set to 115,200 bits per second. The test performed consisted of repeatedly sending predefined arrays of 12 numbers from the computer to the Arduino board via the serial port and then to verify if they were received and parsed correctly and in the same order. Data overflow in the serial port buffer was avoided by introducing a delay of 70.0 ms between each transmission of

the number arrays. The accuracy of the transmitted data was subsequently verified by examining the received serial port data.

The last step of the system testing was to check the performance of the connected system with the human operator. We verified that we were able to operate the H31 quadrotor in a similar manner to the FQ777 helicopter. The flight time was approximately six minutes. During this period, we piloted the drone using the Leap Motion sensor. Since we had greater resolution for the control signals with this design, we observed that we were able to make more precise maneuvers with this drone.

C. CLOSED-LOOP SYSTEM USING WEBCAM AND LEAP MOTION SENSOR

1. Introduction

The main objective of this part of the thesis project was to add a control feedback loop to the architecture of the system depicted in Figure 25. This was needed to achieve more accurate control of position and orientation. The proposed feedback control loop consisted of one webcam that was used to track the position of the quadrotor within the space in 2D. Image processing was used to establish the position of the drone; the control loop was utilized to adjust throttle and roll accordingly. The proposed system architecture is shown in Figure 32.

2. Implementation

To perform the required modifications to the system, we needed to develop a Processing language program to capture and process webcam video that located and tracked the position of the drone within the field-of-view of the camera. Since the Processing software is oriented mainly to image and video applications, the task of capturing video from the webcam was easily implemented.

Next, we needed to utilize an image processing algorithm that provided the ability to detect and track the position of the drone. For this matter, two different options were explored using algorithms develop by Shiffman in [10]:

1. Movement-based tracking algorithm, and

2. Color-based tracking algorithm.

The operating principle of the first algorithm, based on motion, basically examines each of the pixels present in an image frame and compares them with those of the previous image frame. The output value of this algorithm is the mean distance (center of gravity), expressed in (*x*,*y*) coordinates, of all the pixels that change in one iteration.

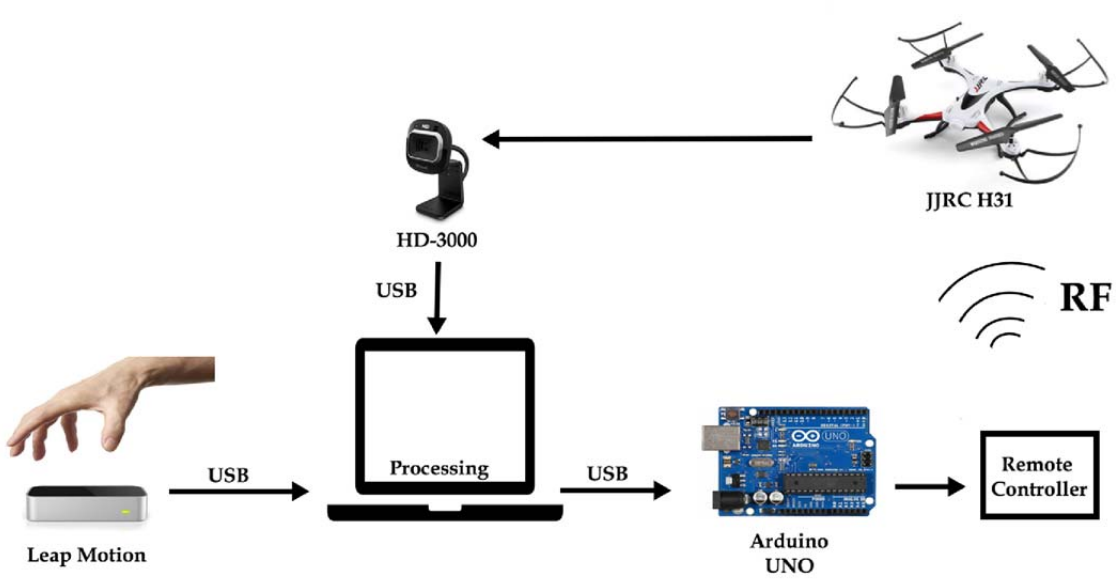

Figure 32. Architecture of the System to Control the JJRC H31 Modified with an Inner Control Loop

The FQ777 helicopter was used to test the tracking algorithm based on motion. We observed that when the helicopter was flying several feet away from the camera, the image of the helicopter was relatively small and the algorithm was able to track it accurately. When the helicopter was closer to the camera, the tracked object within the image increased in size, and the output position of the center of gravity was not stable. Another downside observed during the tests was that if there was a second object moving in the background, the output position of the center of gravity was shifted by some amount towards the direction of the second moving object. A sequence of three image frames showing the operation of the tracking algorithm based on movement is found in

Figure 33. A solid green square marks the center of gravity of the FQ777 helicopter while flying.

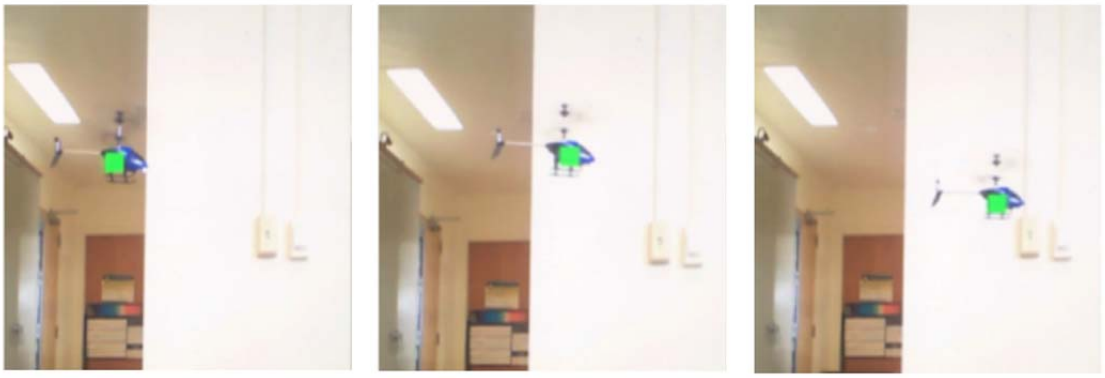

Figure 33. Test of the Tracking Algorithm Based on Movement

The second tracking algorithm, which was based on color, started by analyzing the image frame in search for a specific color region. When it was found, a black circular marker was placed at that location. The algorithm subsequently determined if the color region had changed position and then updated the location of the track.

The H31 quadrotor was used to test the tracking algorithm based on color. The results obtained in the tests of this algorithm were in general much better than those obtained with the movement-tracking algorithm in terms of stability. Background moving objects did not affect the performance of the algorithm provided that they had a different color than the one that was being tracked. From the series of images depicted in Figure 34, we observe the operation of the tracking algorithm based on color. The black circle tracks the position of an orange ball placed on top of the H31 quadrotor. The Processing code for the color-based tracking algorithm is found in Appendix G.

The output of the position tracking algorithm was the (x,y) coordinates of the object located within the image, with the x-coordinate in the horizontal direction and the y-coordinate in the vertical direction. The origin of the image coordinates was set by default in the upper left corner of the image, and every position (x,y) represented a single pixel within the image frame.

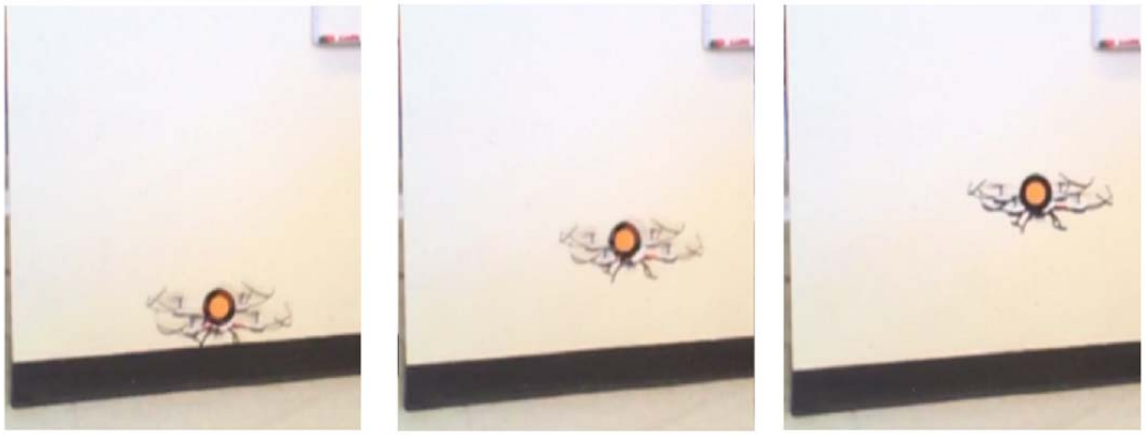

Figure 34. Test of the Algorithm Based on Color

The next step of the experiment was to develop a Processing language program to implement the proposed closed-loop control system. A program was written to track and regulate the 2D position of the drone using one webcam. The computer program controlled the throttle and roll axes of the drone. To accomplish this task, two independent PID controllers (one for each axis) were employed to minimize the error distance between the actual position of the drone and the desired position. The Processing code for drone control using one webcam is found in Appendix H.

The logic behind the Processing language code started when the auto-track function was activated. This initiated the color-tracking algorithm of the drone. When the desired position was entered into the program, the Processing program computed the appropriate throttle and roll commands to minimize the error distance. For this program to work accurately, the drone had to be set to headless mode. When the headless mode was activated on the drone, the onboard flight microcontroller lined up the drone movements to be relative to the remote controller position. A flow chart of the program was developed to operate the H31 drone autonomously with video feed from one webcam and is shown in Figure 35.

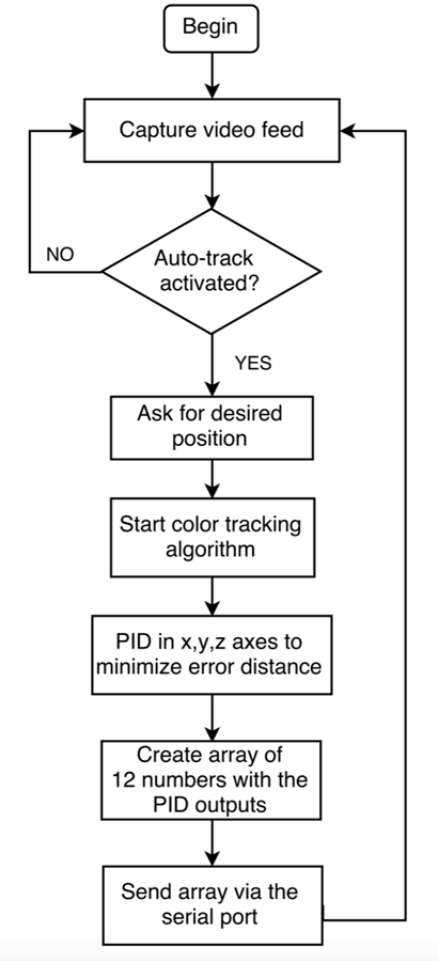

Figure 35. Flow Chart of the Processing Program to Control the H31 with One Webcam

3. System Testing

In order to test the system design shown in Figure 32, we first examined the *y*-axis PID controller, which serves to minimize the vertical error distance. The output of this controller was the throttle command for the H31 drone.

The next step was to test the *x*-axis, which minimizes the horizontal error distance. For this axis, the corresponding PID controller generated the required roll command.

We included some modifications in the original code in order to be able to change in real time the gains of the PID controller. These modifications allowed us to see the

response of the drone to the different values of proportional, integral, and derivative gains while the drone was flying.

During the tests of the system, adjustment of the PID controller gains was very challenging. For this reason, we decided to adopt a heuristic method for tuning PID controllers developed by Ziegler and Nichols [22]. This method required that the integral and derivative gains initially be set to zero, and then we gradually increased the proportional gain until the output of the control loop exhibited a stable oscillation with a period equal to T_u. The last value of the proportional gain is defined as the ultimate gain K_u. The PID gain parameters K_p, T_i, and T_d are obtained from Table 6.

Table 6. Ziegler–Nichols Method. Source: [23].

	K_p	T_i (s)	T_d (s)
P	$0.50K_u$	---	---
PI	$0.45K_u$	$T_u / 1.2$	---
PID	$0.60K_u$	$T_u / 2.0$	$T_u / 8.0$

The output command $u(t)$ needed to control the distance error $e(t)$ was obtained using

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\theta)\,d\theta + T_d\left(\frac{de(t)}{dt}\right)\right). \quad (1)$$

A series of images showing the operation of the control loop are shown in Figure 36. The red "+" sign marks the center of the image, and the red solid square indicates the desired position to which the drone should move. The green square is the result of the color-tracking algorithm following an orange ball placed on top of the drone. In the sequence of pictures, we observe that the system is trying to decrease both vertical and horizontal distance error.

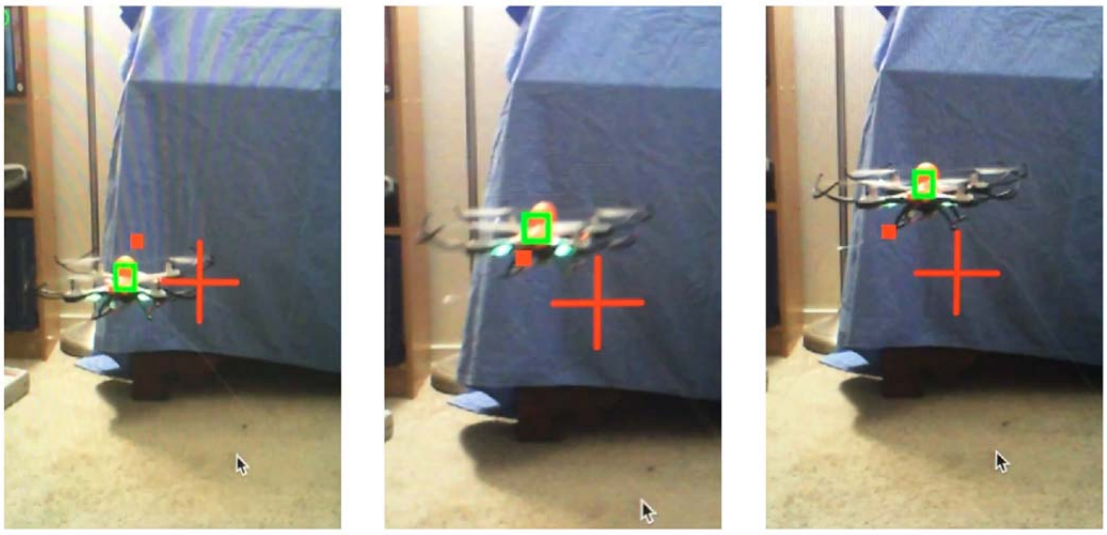

Figure 36. PID Gains Test in X and Y Axes

Having adjusted the PID gains to a nominal set of values using the Ziegler–Nichols method, we still observed oscillations in the drone position for both control channels (throttle and roll). This led us to investigate how fast these corrections were being calculated and transmitted by the program developed for this experiment.

We knew that the Processing main loop was set by default to run at a rate of 60.0 Hz, but the webcam was only capable of transmitting images at a maximum rate of 30.0 Hz. This meant that we were limited by the webcam to run the main control loop at half of the desired rate. Furthermore, we needed to include an additional delay of 70.0 ms to avoid the overflow in the Arduino serial port buffer. This resulted in that the main control loop operated at a rate equal to 7.5 Hz. For this reason, we continued to observe the oscillations in the drone position; the control loop was not fast enough to precisely correct the error distance.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS

A. SUMMARY

The goal of this thesis was to develop an easy-to-use interface to operate a drone. More specifically, the main research objective was to design a simple user interface incorporating the Leap Motion sensor which would allow the drone operator to control a drone using natural hand motions.

This research demonstrated that it was possible to adequately control and operate several examples of commercially-available drones. The motion of the drones was coordinated through the Leap Motion sensor via computer programs that were developed using the Processing programming language. The research showed that the position and orientation of a drone could be controlled in a manner that was natural and intuitive for the operator using this sensor.

The first demonstration of the Leap Motion sensor to operate a drone used the IR FQ777 helicopter. To accomplish this, the IR commands from the remote controller were captured and synthetized using an IR signal capture circuit. A matrix of 50 different commands was created and saved into an Arduino Mega board. A circuit dedicated to transmit IR commands was built and connected to the Arduino Mega microcontroller. A Processing language program was written to serve as a bridge between the Leap Motion sensor and the Arduino based IR signal emitter.

To demonstrate the control of a more advanced drone, the RF JJRC H31 quadrotor, a different method was explored. A digital potentiometer controlled by an Arduino Uno microcontroller was connected directly into the remote controller. This method increased the resolution to 156 different settings in each of the four control channels (throttle, yaw, pitch, and roll). The Processing language computer program was modified to handle one additional channel (roll) as well as the addition of an algorithm in charge of compiling the values of the four channels into one array and send it to the Arduino Uno microcontroller via the serial port.

A closed-loop system consisting of one webcam was created to track and control the position and orientation of the JJRC H31 quadrotor. The tracking algorithm used was based on color. The position obtained from this algorithm was utilized to drive the drone to a desired position. To accomplish this function, two independent PID controllers (one for the vertical axis and one for the horizontal axis) were applied to minimize the error distance between the actual position of the drone and the desired position.

The complete user interface was successfully tested and demonstrated with both the infrared-controlled and the radio frequency-controlled drones utilized in this research. We observed that the user interface made the control of the drones easier and more intuitive because it exploited the natural movements and reactions of the operator's hand.

B. FUTURE RESEARCH

The connection between the desktop computer and the Arduino microcontroller needs to be improved. During the tests performed in this research, we observed an overflow in the serial port buffer that forced us to introduce a delay of 70.0 ms to overcome this problem. A communication channel based on Bluetooth or Wi-Fi can be explored to increase the communication data rate.

Additional investigation into the position tracking capability of the system is also necessary. We proved that the control rate using a webcam connected to the computer running a Processing program was not fast enough to maintain stability of a flying drone. The use of other tracking system like the Prime 17W motion capture system from OptiTrack [24], recently installed in the Controls System Laboratory, provides frame rates up to 360 Hz.

We can take advantage of the dynamics of the hand motion to program the response of the system based on hands' position in space, orientation, and even predefined gestures. In this research work, we proved that it was possible to achieve control of a drone using different positions of the hand with respect of the Cartesian coordinate origin of the Leap Motion sensor, as seen in Figure 37. In [7], the author explored different approaches of controlling a drone based on the orientation of the hand, as seen in Figure 38, and also based on gestures.

The use of predefined gestures opens up a new category of options in the control of drones. Through gestures we can easily execute a large variety of maneuvers, such as landing, takeoff, backflip maneuvers, barrel rolls, bank turns, brake turns, automatic return to base, etc. We can even activate or deactivate sensors on board, execute predefined flying paths, launch missiles, or deploy counter measures.

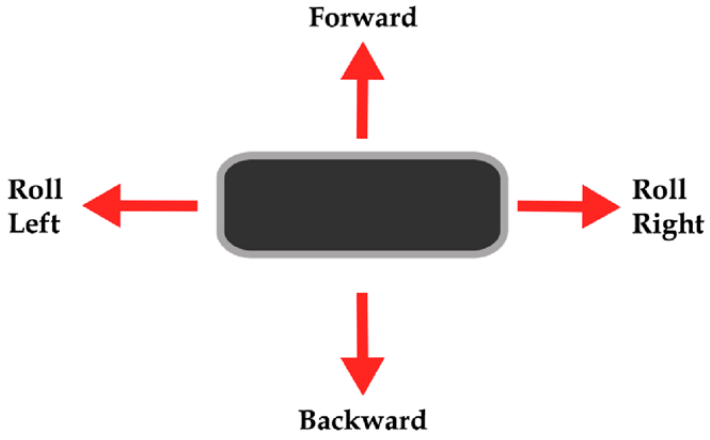

Figure 37. Commands with Respect to the Position of the Hand

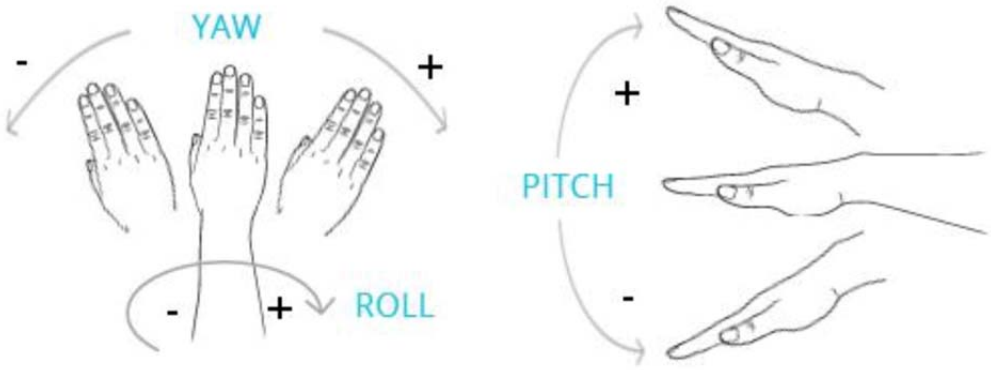

Figure 38. Commands with Respect to the Orientation of the Hand. Source: [7].

Throughout our research, the control of both drones was developed with the use of a single hand, but the sensor features the capability to detect two hands at the same time. This enables the possibility of using a system with a single operator that can apply the movement of his two hands to control two drones at the same time or one drone and some additional on board arrangement (manipulators, sensors, weapons, etc.). Control using two hands can even be used to operate different groups of drones flying in swarms.

APPENDIX A. MATRIX OF IR COMMANDS FOR THE FQ777

	THROTTLE	RIGHT	LEFT	FWD	BACK
1	Q	A	Z	e	o
2	W	S	X	f	p
3	E	D	C	g	q
4	R	F	V	h	r
5	T	G	B	i	s
6	Y	H	N	j	t
7	U	J	M	k	u
8	I	K	b	l	v
9	O	L	c	m	w
10	P	a	d	n	x

Q	800 750 750 800 700 850 300 500 650 500 300 450 650 550 250 500 300 500 250 500 300 500 650 500 300 500 650 500 300 850 300 500 250 900 650 500 650 900 250
A	750 800 700 800 700 850 300 850 300 500 650 500 650 550 250 500 300 450 300 500 650 500 700 500 250 500 650 500 300 850 300 500 300 500 250 500 300 850 650
Z	750 800 700 850 650 900 250 500 300 500 250 500 300 500 300 450 300 500 300 450 300 500 650 500 300 500 650 500 300 850 300 500 250 900 300 450 700 850 650
e	750 800 700 850 650 850 300 500 650 500 300 500 650 500 300 450 300 500 300 450 300 900 250 500 700 850 250 500 300 850 300 500 300 500 250 500 300 500 650
o	800 750 750 800 700 850 300 500 650 500 250 500 700 500 250 500 300 500 250 500 300 500 300 450 300 500 300 450 300 850 300 500 300 850 300 500 650 850 700

W	750 750 750 800 700 850 300 500 650 500 300 450 700 500 300 450 700 450 700 850 300 500 650 500 300 450 700 500 250 900 250 500 300 850 300 500 300 450 300
S	750 800 750 800 700 850 300 850 300 450 700 450 700 500 300 450 700 450 700 850 300 500 650 500 300 450 700 500 300 850 300 450 300 500 650 500 700 450 300
X	750 750 750 800 700 850 300 500 250 500 300 500 250 500 300 500 650 500 650 900 250 500 700 450 300 500 650 500 300 850 300 500 300 850 650 500 300 500 650
f	750 800 700 800 700 850 300 500 650 500 300 500 650 500 300 450 700 500 650 850 300 850 300 500 650 850 300 500 300 850 300 500 250 500 700 500 650 850 650

p	750 800 750 800 700 850 250 500 700 500 250 500 700 450 300 500 650 500 650 900 250 500 300 500 300 450 300 500 300 850 300 500 250 900 650 500 300 450 700

E	750 800 700 800 700 850 300 500 650 500 300 450 700 500 250 500 700 850 650 900 250 500 700 450 300 500 650 500 300 850 300 500 300 850 300 850 300 450 300
D	800 750 750 800 700 850 300 850 300 500 650 500 650 500 300 500 650 850 700 850 300 500 650 500 250 500 700 500 250 900 250 500 300 500 650 850 700 500 250
C	750 800 700 850 650 850 300 500 300 500 250 500 300 500 300 450 700 850 650 850 300 500 700 450 300 500 650 500 300 850 300 500 250 900 650 850 300 500 650
g	750 800 700 850 650 900 250 500 700 450 300 500 650 500 300 500 650 850 700 850 300 850 300 500 650 850 300 500 300 850 300 450 300 500 650 900 650 850 700
q	800 750 750 800 700 850 300 500 650 500 300 450 700 500 250 500 650 900 650 850 300 500 300 500 250 500 300 500 300 850 300 450 300 850 700 850 300 450 700

R	800 750 750 800 700 850 300 500 650 500 300 450 700 500 250 900 250 900 650 500 650 500 700 450 300 500 650 500 300 850 300 450 300 500 700 850 300 850 650
F	750 800 700 850 700 850 300 850 300 450 700 500 650 500 300 850 300 850 650 500 650 500 700 450 300 500 650 500 300 850 300 500 300 850 300 850 650 850 700
V	750 800 700 850 650 850 300 500 300 500 250 500 300 500 300 850 300 850 650 500 250 500 700 500 250 500 700 450 300 850 300 500 300 500 250 900 250 850 700
h	750 800 700 850 700 800 300 500 700 450 300 500 650 500 300 850 300 850 700 450 300 900 250 500 650 900 250 500 300 850 300 500 300 850 300 850 650 500 650
r	800 750 750 800 700 850 300 500 650 500 300 450 700 500 250 900 250 900 650 500 250 500 300 500 300 450 300 500 300 850 300 500 250 500 300 850 300 850 700

T	750 800 700 800 700 850 300 500 650 500 300 500 650 500 250 900 650 850 300 850 700 500 650 500 300 450 700 500 250 900 250 500 300 500 300 850 650 500 650
G	750 800 750 800 700 800 350 850 300 450 700 450 700 500 250 900 650 850 300 850 700 500 250 850 700 850 650 500 300 850 300 500 300 850 300 450 700 850 650
B	750 800 700 850 650 850 300 500 300 500 250 500 300 500 300 850 650 850 300 850 700 500 650 500 300 450 700 500 250 900 250 500 300 500 650 850 700 500 250
i	750 800 700 850 650 850 300 500 650 500 300 500 650 500 300 850 650 900 250 850 700 850 300 500 650 850 300 500 300 850 300 500 250 900 650 850 300 850 300
s	800 750 750 800 700 850 300 500 650 500 300 450 700 500 250 900 650 850 300 850 300 500 300 450 300 500 300 500 250 900 250 500 300 500 650 850 700 500 650

Y	750 800 700 800 700 850 300 500 650 500 300 450 700 500 650 500 300 450 700 850 300 500 650 500 250 500 700 500 250 900 250 500 300 850 700 450 300 500 300
H	800 750 750 800 700 850 300 850 300 500 650 500 650 500 650 500 300 500 650 500 650 500 650 500 300 500 650 500 300 850 300 450 300 500 300 500 650 850 700
N	750 750 750 800 700 850 300 500 250 500 300 500 250 500 700 450 300 500 650 500 650 500 700 500 250 500 650 500 300 850 300 500 300 850 300 450 300 850 300
j	750 800 700 850 650 900 250 500 700 450 300 500 650 500 700 450 300 500 650 500 650 900 300 450 700 850 300 500 250 900 250 500 300 500 300 450 700 450 300
t	750 800 700 850 650 900 250 500 700 450 300 500 650 500 700 450 300 500 650 500 650 500 300 500 300 450 300 500 300 850 300 450 300 900 250 500 300 850 300

U	750 800 700 850 650 850 300 500 650 500 300 500 650 500 650 500 650 500 300 850 700 450 700 500 250 500 650 500 300 850 300 500 300 850 300 450 700 500 650
J	750 800 750 800 650 900 250 900 300 450 700 450 700 500 650 500 650 500 300 850 650 500 300 850 700 850 650 500 300 850 300 500 250 500 300 850 700 850 650
M	750 750 750 800 700 850 300 500 300 450 300 500 300 450 700 500 650 500 300 850 300 450 700 500 250 500 650 500 300 850 300 500 250 900 650 500 650 500 700
k	800 750 750 800 650 900 250 550 650 500 250 500 650 550 650 500 650 500 300 850 300 850 300 450 700 850 300 500 250 900 250 500 300 500 650 500 300 850 650
u	750 800 700 850 650 900 250 500 650 550 250 500 650 500 700 450 700 500 250 500 300 500 250 500 300 500 250 550 250 850 300 500 300 850 650 500 700 850 650

I	750 800 700 850 650 850 300 500 700 450 300 500 650 500 700 450 700 850 300 450 700 500 650 500 300 450 700 500 250 900 250 500 300 850 300 850 700 850 650
K	800 750 750 800 650 900 250 900 300 500 650 500 650 500 650 500 700 850 250 500 700 500 650 500 300 450 700 500 250 850 300 500 300 500 650 850 300 850 700
b	750 800 750 750 750 800 300 500 300 450 300 500 300 500 650 500 650 850 300 500 650 500 650 500 300 500 650 500 300 850 300 450 300 900 650 850 700 850 300
l	750 800 750 750 750 800 350 450 700 450 300 500 650 500 700 450 700 850 300 450 700 850 300 500 650 850 300 500 300 850 300 450 300 500 700 850 250 500 300
v	750 750 750 800 700 850 300 500 650 500 300 450 700 500 650 500 650 850 300 500 650 500 300 500 250 500 300 500 300 850 300 450 300 850 700 850 700 850 250

O	750 800 700 850 700 850 250 500 700 500 250 500 700 450 700 850 300 850 650 500 650 550 650 500 250 500 650 550 250 850 300 500 300 500 650 850 300 850 650
L	750 800 700 850 650 900 250 900 250 500 700 450 700 500 650 850 300 850 700 450 300 500 650 500 300 500 650 500 300 850 300 450 300 900 250 900 650 850 300
c	750 800 700 850 650 850 300 500 300 500 250 500 300 500 650 850 300 850 700 450 300 500 700 450 300 500 650 500 300 850 300 450 300 500 300 850 300 850 650
m	750 800 700 850 650 900 250 500 700 450 300 500 650 500 700 850 250 900 650 500 300 850 300 500 650 850 300 500 250 900 300 450 300 850 300 850 700 450 700
w	750 750 750 800 700 850 300 500 650 500 300 450 700 500 650 850 300 850 700 450 300 500 300 500 250 500 300 500 300 850 300 450 300 500 300 850 300 850 650

P	800 750 750 800 700 850 300 500 650 500 300 450 700 850 300 500 250 500 650 500 650 550 650 500 250 500 700 500 250 850 300 500 300 850 650 500 300 850 650
a	750 800 700 850 650 900 250 900 300 450 700 450 700 850 300 500 250 500 700 450 700 500 650 500 300 450 700 500 250 900 250 500 300 500 300 450 700 850 650
d	800 750 750 800 700 850 300 500 250 500 300 450 300 900 300 450 300 500 650 500 650 500 650 500 300 500 650 500 300 850 300 500 250 900 300 450 300 850 300
n	750 800 700 800 700 850 300 500 650 500 300 500 650 850 300 500 300 450 700 450 700 850 300 500 650 850 300 500 300 850 300 450 300 500 300 500 650 500 250
x	750 800 750 800 700 850 300 500 650 500 300 450 700 850 300 500 250 500 650 500 700 500 250 500 300 450 300 500 300 850 300 500 250 900 300 450 300 850 300

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. ARDUINO CODE FOR IR EMITTER

```
#include <IRremote.h>

int IRLED = 9; //in UNO is pin 3, in MEGA is pin 9
decode_results results;
IRsend irsend;

unsigned int letters[51][39] =
{
        {750, 800, 700, 800, 700, 850, 300, 850, 300, 500, 650, 500, 650, 550, 250, 500,
        300, 450, 300, 500, 650, 500, 700, 500, 250, 500, 650, 500, 300, 850, 300, 500,
        300, 500, 250, 500, 300, 850, 650},
        {750, 800, 700, 850, 650, 850, 300, 500, 300, 500, 250, 500, 300, 500, 300, 850,
        650, 850, 300, 850, 700, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 500,
        300, 500, 650, 850, 700, 500, 250},
        {750, 800, 700, 850, 650, 850, 300, 500, 300, 500, 250, 500, 300, 500, 300, 450,
        700, 850, 650, 850, 300, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 500,
        250, 900, 650, 850, 300, 500, 650},
        {800, 750, 750, 800, 700, 850, 300, 850, 300, 500, 650, 500, 650, 500, 300, 500,
        650, 850, 700, 850, 300, 500, 650, 500, 250, 500, 700, 500, 250, 900, 250, 500,
        300, 500, 650, 850, 700, 500, 250},
        {750, 800, 700, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 500,
        700, 850, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 500,
        300, 850, 300, 850, 300, 450, 300},
        {750, 800, 700, 850, 700, 850, 300, 850, 300, 450, 700, 500, 650, 500, 300, 850,
        300, 850, 650, 500, 650, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 500,
        300, 850, 300, 850, 650, 850, 700},
        {750, 800, 750, 800, 700, 800, 350, 850, 300, 450, 700, 450, 700, 500, 250, 900,
        650, 850, 300, 850, 700, 500, 250, 850, 700, 850, 650, 500, 300, 850, 300, 500,
        300, 850, 300, 450, 700, 850, 650},
        {800, 750, 750, 800, 700, 850, 300, 850, 300, 500, 650, 500, 650, 500, 650, 500,
        300, 500, 650, 500, 650, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 450,
        300, 500, 300, 500, 650, 850, 700},
        {750, 800, 700, 850, 650, 850, 300, 500, 700, 450, 300, 500, 650, 500, 700, 450,
        700, 850, 300, 450, 700, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 500,
        300, 850, 300, 850, 700, 850, 650},
        {750, 800, 750, 800, 650, 900, 250, 900, 300, 450, 700, 450, 700, 500, 650, 500,
        650, 500, 300, 850, 650, 500, 300, 850, 700, 850, 650, 500, 300, 850, 300, 500,
        250, 500, 300, 850, 700, 850, 650},
        {800, 750, 750, 800, 650, 900, 250, 900, 300, 500, 650, 500, 650, 500, 650, 500,
        700, 850, 250, 500, 700, 500, 650, 500, 300, 450, 700, 500, 250, 850, 300, 500,
        300, 500, 650, 850, 300, 850, 700},
```

{750, 800, 700, 850, 650, 900, 250, 900, 250, 500, 700, 450, 700, 500, 650, 850, 300, 850, 700, 450, 300, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 450, 300, 900, 250, 900, 650, 850, 300},
{750, 750, 750, 800, 700, 850, 300, 500, 300, 450, 300, 500, 300, 450, 700, 500, 650, 500, 300, 850, 300, 450, 700, 500, 250, 500, 650, 500, 300, 850, 300, 500, 250, 900, 650, 500, 650, 500, 700},
{750, 750, 750, 800, 700, 850, 300, 500, 250, 500, 300, 500, 250, 500, 700, 450, 300, 500, 650, 500, 650, 500, 700, 500, 250, 500, 650, 500, 300, 850, 300, 500, 300, 850, 300, 450, 300, 850, 300},
{750, 800, 700, 850, 700, 850, 250, 500, 700, 500, 250, 500, 700, 450, 700, 850, 300, 850, 650, 500, 650, 550, 650, 500, 250, 500, 650, 550, 250, 850, 300, 500, 300, 500, 650, 850, 300, 850, 650},
{800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 850, 300, 500, 250, 500, 650, 500, 650, 550, 650, 500, 250, 500, 700, 500, 250, 850, 300, 500, 300, 850, 650, 500, 300, 850, 650},
{800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 650, 550, 250, 500, 300, 500, 250, 500, 300, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 500, 250, 900, 650, 500, 650, 900, 250},
{800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 900, 650, 500, 650, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 450, 300, 500, 700, 850, 300, 850, 650},
{750, 800, 750, 800, 700, 850, 300, 850, 300, 450, 700, 450, 700, 500, 300, 450, 700, 450, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 300, 850, 300, 450, 300, 500, 650, 500, 700, 450, 300},
{750, 800, 700, 800, 700, 850, 300, 500, 650, 500, 300, 500, 650, 500, 250, 900, 650, 850, 300, 850, 700, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 500, 300, 500, 300, 850, 650, 500, 650},
{750, 800, 700, 850, 650, 850, 300, 500, 650, 500, 300, 500, 650, 500, 650, 500, 650, 500, 300, 850, 700, 450, 700, 500, 250, 500, 650, 500, 300, 850, 300, 500, 300, 850, 300, 450, 700, 500, 650},
{750, 800, 700, 850, 650, 850, 300, 500, 300, 500, 250, 500, 300, 500, 300, 850, 300, 850, 650, 500, 250, 500, 700, 500, 250, 500, 700, 450, 300, 850, 300, 500, 300, 500, 250, 900, 250, 850, 700},
{750, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 300, 450, 700, 450, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 500, 300, 850, 300, 500, 300, 450, 300},
{750, 750, 750, 800, 700, 850, 300, 500, 250, 500, 300, 500, 250, 500, 300, 500, 650, 500, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 500, 300, 850, 650, 500, 300, 500, 650},
{750, 800, 700, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 650, 500, 300, 450, 700, 850, 300, 500, 650, 500, 250, 500, 700, 500, 250, 900, 250, 500, 300, 850, 700, 450, 300, 500, 300},
{750, 800, 700, 850, 650, 900, 250, 500, 300, 500, 250, 500, 300, 500, 300, 450, 300, 500, 300, 450, 300, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 500, 250, 900, 300, 450, 700, 850, 650},

{750, 800, 700, 850, 650, 900, 250, 900, 300, 450, 700, 450, 700, 850, 300, 500, 250, 500, 700, 450, 700, 500, 650, 500, 300, 450, 700, 500, 250, 900, 250, 500, 300, 500, 300, 450, 700, 850, 650},
{750, 800, 750, 750, 750, 800, 300, 500, 300, 450, 300, 500, 300, 500, 650, 500, 650, 850, 300, 500, 650, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 450, 300, 900, 650, 850, 700, 850, 300},
{750, 800, 700, 850, 650, 850, 300, 500, 300, 500, 250, 500, 300, 500, 650, 850, 300, 850, 700, 450, 300, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 450, 300, 500, 300, 850, 300, 850, 650},
{800, 750, 750, 800, 700, 850, 300, 500, 250, 500, 300, 450, 300, 900, 300, 450, 300, 500, 650, 500, 650, 500, 650, 500, 300, 500, 650, 500, 300, 850, 300, 500, 250, 900, 300, 450, 300, 850, 300},
{750, 800, 700, 850, 650, 850, 300, 500, 650, 500, 300, 500, 650, 500, 300, 450, 300, 500, 300, 450, 300, 900, 250, 500, 700, 850, 250, 500, 300, 850, 300, 500, 300, 500, 250, 500, 300, 500, 650},
{750, 800, 700, 800, 700, 850, 300, 500, 650, 500, 300, 500, 650, 500, 300, 450, 700, 500, 650, 850, 300, 850, 300, 500, 650, 850, 300, 500, 300, 850, 300, 500, 250, 500, 700, 500, 650, 850, 650},
{750, 800, 700, 850, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 300, 500, 650, 850, 700, 850, 300, 850, 300, 500, 650, 850, 300, 500, 300, 850, 300, 450, 300, 500, 650, 900, 650, 850, 700},
{750, 800, 700, 850, 700, 800, 300, 500, 700, 450, 300, 500, 650, 500, 300, 850, 300, 850, 700, 450, 300, 900, 250, 500, 650, 900, 250, 500, 300, 850, 300, 500, 300, 850, 300, 850, 650, 500, 650},
{750, 800, 700, 850, 650, 850, 300, 500, 650, 500, 300, 500, 650, 500, 300, 850, 650, 900, 250, 850, 700, 850, 300, 500, 650, 850, 300, 500, 300, 850, 300, 500, 250, 900, 650, 850, 300, 850, 300},
{750, 800, 700, 850, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 700, 450, 300, 500, 650, 500, 650, 900, 300, 450, 700, 850, 300, 500, 250, 900, 250, 500, 300, 500, 300, 450, 700, 450, 300},
{800, 750, 750, 800, 650, 900, 250, 550, 650, 500, 250, 500, 650, 550, 650, 500, 650, 500, 300, 850, 300, 850, 300, 450, 700, 850, 300, 500, 250, 900, 250, 500, 300, 500, 650, 500, 300, 850, 650},
{750, 800, 750, 750, 750, 800, 350, 450, 700, 450, 300, 500, 650, 500, 700, 450, 700, 850, 300, 450, 700, 850, 300, 500, 650, 850, 300, 500, 300, 850, 300, 450, 300, 500, 700, 850, 250, 500, 300},
{750, 800, 700, 850, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 700, 850, 250, 900, 650, 500, 300, 850, 300, 500, 650, 850, 300, 500, 250, 900, 300, 450, 300, 850, 300, 850, 700, 450, 700},
{750, 800, 700, 800, 700, 850, 300, 500, 650, 500, 300, 500, 650, 850, 300, 500, 300, 450, 700, 450, 700, 850, 300, 500, 650, 850, 300, 500, 300, 850, 300, 450, 300, 500, 300, 500, 650, 500, 250},
{800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 250, 500, 700, 500, 250, 500, 300, 500, 250, 500, 300, 500, 300, 450, 300, 500, 300, 450, 300, 850, 300, 500, 300, 850, 300, 500, 650, 850, 700},

```
        {750, 800, 750, 800, 700, 850, 250, 500, 700, 500, 250, 500, 700, 450, 300, 500,
        650, 500, 650, 900, 250, 500, 300, 500, 300, 450, 300, 500, 300, 850, 300, 500,
        250, 900, 650, 500, 300, 450, 700},
        {800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 500,
        650, 900, 650, 850, 300, 500, 300, 500, 250, 500, 300, 500, 300, 850, 300, 450,
        300, 850, 700, 850, 300, 450, 700},
        {800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 900,
        250, 900, 650, 500, 250, 500, 300, 500, 300, 450, 300, 500, 300, 850, 300, 500,
        250, 500, 300, 850, 300, 850, 700},
        {800, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 250, 900,
        650, 850, 300, 850, 300, 500, 300, 450, 300, 500, 300, 500, 250, 900, 250, 500,
        300, 500, 650, 850, 700, 500, 650},
        {750, 800, 700, 850, 650, 900, 250, 500, 700, 450, 300, 500, 650, 500, 700, 450,
        300, 500, 650, 500, 650, 500, 300, 500, 300, 450, 300, 500, 300, 850, 300, 450,
        300, 900, 250, 500, 300, 850, 300},
        {750, 800, 700, 850, 650, 900, 250, 500, 650, 550, 250, 500, 650, 500, 700, 450,
        700, 500, 250, 500, 300, 500, 250, 500, 300, 500, 250, 550, 250, 850, 300, 500,
        300, 850, 650, 500, 700, 850, 650},
        {750, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 650, 500,
        650, 850, 300, 500, 650, 500, 300, 500, 250, 500, 300, 500, 300, 850, 300, 450,
        300, 850, 700, 850, 700, 850, 250},
        {750, 750, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 500, 650, 850,
        300, 850, 700, 450, 300, 500, 300, 500, 250, 500, 300, 500, 300, 850, 300, 450,
        300, 500, 300, 850, 300, 850, 650},
        {750, 800, 750, 800, 700, 850, 300, 500, 650, 500, 300, 450, 700, 850, 300, 500,
        250, 500, 650, 500, 700, 500, 250, 500, 300, 450, 300, 500, 300, 850, 300, 500,
        250, 900, 300, 450, 300, 850, 300}
};

void setup()
{
        Serial.begin(115200);
        pinMode(IRLED, OUTPUT);
}

void slow_down()
{
        irsend.sendRaw(letters[20],78,38); //U
        irsend.sendRaw(letters[22],78,38); //Y
        irsend.sendRaw(letters[19],78,38); //T
        irsend.sendRaw(letters[17],78,38); //R
        irsend.sendRaw(letters[4],78,38);   //E
        irsend.sendRaw(letters[22],78,38); //W
}
```

```
void loop()
{
	if (Serial.available() > 0)
	{
		char letter_to_send = Serial.read();
		if (letter_to_send == ‘#’){
			slow_down();
		}
		if (isLowerCase(letter_to_send))
		{
			char letters_index = letter_to_send - ‘a’ + 26;
			irsend.sendRaw(letters[letters_index],sizeof(letters[letters_index]),
			38);
			delay(50);
		}
		else{
			char letters_index = letter_to_send - ‘A’;
			irsend.sendRaw(letters[letters_index],sizeof(letters[letters_index]),
			38);
			delay(50);
		}
	}
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. PROCESSING CODE FOR LEAP MOTION TEST

```
import de.voidplus.leapmotion.*;

LeapMotion leap;

float x, y, z;

void setup()
{
        background(255);
        leap = new LeapMotion(this);
}

void draw()
{
        background(255);
        for (Hand hand : leap.getHands ())
        {
                PVector handPosition   = hand.getPosition();

                x = handPosition.x;

                 y = handPosition.y;

                 z = handPosition.z;

                 print ("X:  " + x + "\t" + "Y:  " + y + "\t" + "Z:  " + z + "\n");
        }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. PROCESSING CODE FOR LEAP MOTION DATA COLLECTION

```
import de.voidplus.leapmotion.*;

LeapMotion leap;
int x, y, z;

void setup()
{
        background(255);
        leap = new LeapMotion(this);
}

void draw()
{
       background(255);
       for (Hand hand : leap.getHands ())
       {
              PVector handPosition   = hand.getPosition();

              //Right and Left from -10 to +10
              x =  int(map(handPosition.x, 1, 100, -10, 10));

              //Up and Down from 1 to 10
               y  =  int(map(handPosition.y, 80, 55, 1, 10));

              //Forward and Back from 1 to 10
               z  =  int(map(handPosition.z, 1, 85, 10, -10));

              print ("X:  " + x + "\t" + "Y:  " + y + "\t" + "Z:  " + z + "\n");
       }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. PROCESSING CODE FOR LEAP MOTION HELO CONTROL

```
import de.voidplus.leapmotion.*;
import processing.serial.*;
Serial myPort;

LeapMotion leap;
int xx, yy, zz;

void setup()
{
        background (255);
        leap = new LeapMotion(this);
        myPort = new Serial(this, "/dev/tty.usbmodem1421," 115200);
        stroke(0);
}

void draw()
{
            background(255);
            for (Hand hand : leap.getHands ())
                {
            PVector handPosition   = hand.getPosition();

            //Right and Left from -10 to +10
            xx= int(map(handPosition.x, 1, 100, -10, 10));

            //Up and Down from 1 to 10
            yy = int(map(handPosition.y, 80, 55, 1, 10));

            //Forward and Back from 1 to 10
            zz = int(map(handPosition.z, 1, 85, 10, -10));

            if (leap.countHands()<1)
                {
                xx=yy=zz=0;
            }

            //10 codes for center power 'Q,W,E,R,T,yy,U,I,O,P'
            if (yy==1 && xx<4 && xx>-3 && zz<6 && zz>-5) {
            myPort.write(81);
            delay(500);
            }
```

```
if (yy==2 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(87);
  delay(500);
}
if (yy==3 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(69);
  delay(500);
}
if (yy==4 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(82);
  delay(500);
}
if (yy==5 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(84);
  delay(500);
}
if (yy==6 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(89);
  delay(500);
}
if (yy==7 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(85);
  delay(500);
}
if (yy==8 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(73);
  delay(500);
}
if (yy==9 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(79);
  delay(500);
}
if (yy>=10 && xx<4 && xx>-3 && zz<6 && zz>-5) {
  myPort.write(80);
  delay(500);


//10 codes for right ‘A,S,D,F,G,H,J,K,L,a’
}
if (yy == 1 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(65);
  delay(500);
}
if (yy == 2 && xx >= 4 && zz < 6 && zz >-4) {
```

```
  myPort.write(83);
  delay(500);
}
if (yy == 3 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(68);
  delay(500);
}
if (yy == 4 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(70);
  delay(500);
}
if (yy == 5 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(71);
  delay(500);
}
if (yy == 6 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(72);
  delay(500);
}
if (yy == 7 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(74);
  delay(500);
}
if (yy == 8 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(75);
  delay(500);
}
if (yy == 9 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(76);
  delay(500);
}
if (yy >= 10 && xx >= 4 && zz < 6 && zz >-4) {
  myPort.write(97);
  delay(500);


//10 codes for left ‘zz,xx,C,V,B,N,M,b,c,d’
}
if (yy == 1 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(90);
  delay(500);
}
if (yy == 2 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(88);
  delay(500);
```

```
}
if (yy == 3 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(67);
  delay(500);
}
if (yy == 4 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(86);
  delay(500);
}
if (yy == 5 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(66);
  delay(500);
}
if (yy == 6 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(78);
  delay(500);
}
if (yy == 7 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(77);
  delay(500);
}
if (yy == 8 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(98);
  delay(500);
}
if (yy == 9 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(99);
  delay(500);
}
if (yy >= 10 && xx <= -4 && zz < 6 && zz >-4) {
  myPort.write(100);
  delay(500);


//10 codes for forward ‘e,f,g,h,i,j,k,l,m,n’
}
if (yy == 1 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(101);
  delay(500);
}
if (yy == 2 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(102);
  delay(500);
}
if (yy == 3 && xx > -4 && xx < 4 && zz <= -5) {
```

```
  myPort.write(103);
  delay(500);
}
if (yy == 4 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(104);
  delay(500);
}
if (yy == 5 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(105);
  delay(500);
}
if (yy == 6 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(106);
  delay(500);
}
if (yy == 7 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(107);
  delay(500);
}
if (yy == 8 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(108);
  delay(500);
}
if (yy == 9 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(109);
  delay(500);
}
if (yy >= 10 && xx > -4 && xx < 4 && zz <= -5) {
  myPort.write(110);
  delay(500);


//10 codes for back ‘o,p,q,r,s,t,u,v,w,xx’
}
if (yy == 1 && xx > -4 && xx < 4 && zz >=6) {
  myPort.write(111);
  delay(500);
}
if (yy == 2 && xx > -4 && xx < 4 && zz >=6) {
  myPort.write(112);
  delay(500);
}
if (yy == 3 && xx > -4 && xx < 4 && zz >=6) {
  myPort.write(113);
  delay(500);
```

```
            }
            if (yy == 4 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(114);
              delay(500);
            }
            if (yy == 5 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(115);
              delay(500);
            }
            if (yy == 6 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(116);
              delay(500);
            }
            if (yy == 7 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(117);
              delay(500);
            }
            if (yy == 8 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(118);
              delay(500);
            }
            if (yy == 9 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write(119);
              delay(500);
            }
            if (yy >= 11 && xx > -4 && xx < 4 && zz >=6) {
              myPort.write (120);
              delay(500);
            }
          }
  myPort.clear();
}
```

APPENDIX F. ARDUINO CODE FOR DIGITAL POTENTIOMETER

```
#include <SPI.h>

const int slaveSelectPin = 10;
const int shutDownPin = 7;
int buffer [14];
int val;

int Throttle, Roll, Yaw, Pitch;


void setup()
{
        pinMode (slaveSelectPin, OUTPUT);
        pinMode (shutDownPin, OUTPUT);
        SPI.begin();
        digitalWrite(shutDownPin, HIGH);
        Serial.begin(115200);
        digitalPotWrite(0, 0);
        digitalPotWrite(1, 75);
        digitalPotWrite(2, 77);
        digitalPotWrite(3, 77);
}

void loop()
{
        if (Serial.available() > 0)
        {
                for (int i = 0; i < 14; i++) {
                buffer[i] = Serial.read() - '0';
                delay(3);
                }
                if (buffer[0] == -13)
                {
                        activateDrone();
                        Serial.println("Activating Drone");
                }
                if (buffer[0] == -7)
                {
                        Calibrate();
                        Serial.println("Calibrating Sensors");
                }
```

```
                if (buffer[0] >= 0 && buffer[0]<=9 && buffer[1] >= 0 && buffer[1] <= 9
                &&
                buffer[2] >= 0 && buffer[2] <= 9 && buffer[3] >= 0 && buffer[3] <= 9
                &&
                buffer[4] >= 0 && buffer[4] <= 9 && buffer[5] >= 0 && buffer[5] <= 9
                &&
                buffer[6] >= 0 && buffer[6] <= 9 && buffer[7] >= 0 && buffer[7] <= 9
                &&
                buffer[8] >= 0 && buffer[8] <= 9 && buffer[9] >= 0 && buffer[9] <= 9
                &&
                buffer[10] >= 0 && buffer[10] <= 9 && buffer[11] >= 0 && buffer[11]
                <= 9)
                {
                        Throttle  = 100 * buffer[0] + 10 * buffer[1]   + buffer[2];
                        Yaw       = 100 * buffer[3] + 10 * buffer[4]   + buffer[5];
                        Pitch     =  100 * buffer[6] + 10 * buffer[7]  + buffer[8];
                        Roll      =  100 * buffer[9] + 10 * buffer[10] + buffer[11];
                }
        }
        Serial.flush();
        if (Throttle >= 155)  Throttle = 155;
        if (Roll >= 156)      Roll     = 156;
        if (Pitch >= 156)     Pitch    = 156;
        if (Yaw >= 154)       Yaw      = 154;
        digitalPotWrite(0, Throttle);
        digitalPotWrite(1, Yaw);
        digitalPotWrite(2, Pitch);
        digitalPotWrite(3, Roll);
}

void digitalPotWrite(int address, int value)
{
        digitalWrite(slaveSelectPin, LOW);
        SPI.transfer(address);
        SPI.transfer(value);
        digitalWrite(slaveSelectPin, HIGH);
}
```

```
void activateDrone()   //  “#”
{
        for (int i = 0; i <= 155; i++) {
                digitalPotWrite(0, i);
                delay(15);
        }
        for (int ii = 0; ii < 155 ; ii++)
        {
                digitalPotWrite(0, 155 - ii);
                delay(15);
        }
}

void Calibrate()        //    “)”
{
        digitalPotWrite(0, 0);
        digitalPotWrite(1, 0);
        digitalPotWrite(2, 156);
        digitalPotWrite(3, 156);
        delay(3000);
        digitalPotWrite(0, 0);
        digitalPotWrite(1, 75);
        digitalPotWrite(2, 77);
        digitalPotWrite(3, 77);
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. PROCESSING CODE FOR COLOR TRACKING

```
// Learning Processing - Daniel Shiffman
// http://www.learningprocessing.com

import processing.video.*;

Capture video;
color trackColor;

void setup()
{
          size(320, 240);
          video = new Capture(this, width, height);
          video.start();
          trackColor = color(255, 0, 0);
}

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

void draw()
{
        video.loadPixels();
        image(video, 0, 0);
        float worldRecord = 500;

        int closestX = 0;
        int closestY = 0;

        for (int x = 0; x < video.width; x ++ )
        {
                 for (int y = 0; y < video.height; y ++ )
                {
                        int loc = x + y*video.width;
                        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 = dist(r1, g1, b1, r2, g2, b2);
                        if (d < worldRecord)
                        {
                                worldRecord = d;
                                closestX = x;
                                closestY = y;
                        }
                }
        }

        if (worldRecord < 10)
        {
                fill(trackColor);
                strokeWeight(4.0);
                stroke(0);
                ellipse(closestX, closestY, 16, 16);
        }
}

void mousePressed()
{
        int loc = mouseX + mouseY*video.width;
        trackColor = video.pixels[loc];
}
```

APPENDIX H. PROCESSING CODE FOR DRONE CONTROL WITH ONE WEBCAM

```
import processing.video.*;
import processing.serial.*;

Capture video;
Serial myPort;

color trackColor=1;
float threshold = 30;              // color threshold
float distThreshold = 35;          // distance between two blobs
float Ltargetsize= 6500;           // Largest target detected
float Stargetsize= 10;             // Smallest target detected


ArrayList<Blob> blobs = new ArrayList<Blob>();
boolean recording = false;
boolean trackingON = false;
boolean red_target_window=false;

float scopeSize=820;

float setX=320;
float setY=400;

String numberToSend;
String dos="075";
String tres="077";
String aa1,aa2,aaToSend;

float lastTime;
float OutputT = 65;
float errSumT, lastErrorT;
float KpT = 0.03;
float KiT = 0.04;
float KdT = 0.031;

Float baseSpeed=70.0;
Float lastSpeed=0.0;
int maxThrottle = 155;
float SampleTime = 100;

float OutputR = 77;
```

```
float errSumR, lastErrorR;
float KpR = 0.03;
float KiR = 0.0;
float KdR = 0.0;
Float baseRoll=77.0;
int maxRoll = 156;

void setup()
{
        size(640, 400);
        myPort = new Serial(this, “/dev/tty.usbmodem14241,” 115200);
        video = new Capture (this, 640, 400, “Microsoft® LifeCam HD-3000 #2,” 30);
        video.start();
        trackColor = color(255, 0, 0);
}

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

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 == ‘g’ || key == ‘G’) KpT = KpT + 0.01;
        if (key == ‘b’ || key == ‘B’) KpT = KpT - 0.01;
         if (key == ‘h’ || key == ‘H’) KiT = KiT + 0.001;
         if (key == ‘n’ || key == ‘N’) KiT = KiT - 0.001;
         if (key == ‘j’ || key == ‘J’) KdT = KdT + 0.001;
         if (key == ‘m’ || key == ‘M’) KdT = KdT - 0.001;
        if (key == ‘k’ || key == ‘K’)   baseSpeed = baseSpeed + 1.0;
        if (key == ‘,’)                         baseSpeed = baseSpeed - 1.0;
        if (key == ‘o’)                 myPort.write(35);

        if (KpT<=0)KpT=0;
        if (KiT<=0)KiT=0;
        if (KdT<=0)KdT=0;

        if ((keyCode == UP))   setY-=10;
        if ((keyCode == DOWN)) setY+=10;
        if ((keyCode == LEFT)) setX-=10;
        if ((keyCode == RIGHT)) setX+=10;
        if (setX>=width) setX=width;
        if (setX<=0)     setX=0;
        if (setY>=height) setY=height;
        if (setY<=0)     setY=0;


        if (key == ‘P’ || key == ‘p’) { // give or take control to auto-traking
                trackingON =! trackingON;
                if (!trackingON) scopeSize=820;
        }
}

void draw()
{
        float ping = millis();
        video.loadPixels();
        image(video, 0, 0);
        blobs.clear();

        for (int x = 0; x < video.width; x++ ) {
                for (int y = 0; y < video.height; y++ ) {
                int loc = x + y * video.width;
                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)) {
         float [] delta=b.show();

//=PID CONTROL

    if (trackingON) {
         float now = millis();
         float timeChange = (now - lastTime);
                 if (timeChange >= SampleTime) {
//===PID THROTTTLE
         float errorT = setY - delta[1];
         errSumT += errorT*KiT;
         if (errSumT > maxThrottle) {
                 errSumT = maxThrottle;
         } else if (errSumT<=0) {
                 errSumT=0;
         }
         float dInputT = (errorT - lastErrorT);

         if (errorT<0) {
                 OutputT = baseSpeed + (-1)*(KpT * errorT + errSumT - KdT * dInputT);
```

```
        } else {
                OutputT = baseSpeed -1;
        }

        lastErrorT = errorT;

        if (OutputT<=0) OutputT=0;
        if (OutputT>=155) OutputT=155;

  //=PID ROLL
      float errorR = setX - delta[0];
      errSumR += errorR*KiR;
      if (errSumR > maxRoll) {
                errSumR = maxRoll;
      } else if (errSumR<=0) {
                errSumR=0;
      }
      float dInputR = (errorR - lastErrorR);

      OutputR = baseRoll + (KpR * errorR + errSumR - KdR * dInputR);

      if (OutputR<=0) OutputR=0;
      if (OutputR>=156) OutputR=156;

      lastErrorR = errorR;
      lastTime = now;
      lastSpeed = OutputT;

     }
    }
    else{
        aa1 = nf(int(lastSpeed),3);
        aa2 = "075077077";
        aaToSend = aa1 + aa2;
        myPort.write(aaToSend);
        delay(100);
    }


print("PosX\t"+int(delta[0])+"\tPosY\t"+int(delta[1])+"\tSetX"+"\t"+setX+"\tSetY"+"\t"
+setY+"\tbaseSpeed\t"+baseSpeed+"\n");
print("KpT\t"+KpT+"\tKiT\t"+KiT+"\tKdT\t"+KdT+"\tErrorT\t"+lastErrorT+"\n");
print("KpR\t"+KpR+"\tKiR\t"+KiR+"\tKdR\t"+KdR+"\tErrorR\t"+lastErrorR+"\n");
```

```
        String a1 = nf(int(OutputT), 3);
        String a4 = nf(int(OutputR), 3);

        numberToSend=a1 + dos + tres + a4;

        print("Output\t" + numberToSend + "\n");

        myPort.write(numberToSend);
        }
}


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);

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);

smooth();
noStroke();
fill(255, 0, 0);
rect(setX, setY, 10, 10);

//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);
  }
  delay(100);
  float pong = millis();
  //println("Loop time = " + (pong-ping));
}

float distSq(float x1, float y1, float x2, float y2) {
          float d = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1);
          return d;
}

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];
}

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};
    }

    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;
        }
    }
}
```

LIST OF REFERENCES

[1] J. Whitehead, “Rise of the drones: Managing the unique risks associated with unmanned aircraft systems,” Allianz Global Co. & Sp., Munich, Germany, 2016. [Online]. Available: https://www.agcs.allianz.com/assets/PDFs/Reports/AGCS_Rise_of_the_drones_report.pdf

[2] L. R. García Carrillo, A. E. Dzul Lopez, R. Lozano, and C. Pégard, *Quad Rotorcraft Control: Vision-Based Hovering and Navigation*. New York, NY, USA: Springer Publishing, 2012.

[3] A. R. Jha, *Theory, Design, and Applications of Unmanned Aerial Vehicles.* Boca Raton, FL, USA: CRC Press, 2016.

[4] P. Grier, “Drone aircraft in a stepped-up war in Afghanistan and Pakistan,” *The Christian Science Monitor*, December 11, 2009. [Online]. Available: https://www.csmonitor.com/USA/Military/2009/1211/Drone-aircraft-in-a-stepped-up-war-in-Afghanistan-and-Pakistan

[5] R. Austin, *Unmanned Aircraft Systems: UAV Design, Development and Deployment*. Hoboken, NJ, USA: John Wiley & Sons, 2010.

[6] “Leap Motion Developer,” Leap Motion. Accessed September 20, 2017. [Online]. Available: https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Overview.html

[7] L. Shao, “Hand movement and gesture recognition using Leap Motion Controller,” *Virtual Reality, Course Report*, Stanford, CA, USA. [Online]. Available: https://stanford.edu/class/ee267/Spring2016/report_lin.pdf

[8] “What is Arduino?” Arduino. Accessed September 20, 2017. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[9] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*. Cambridge, MA, USA: The MIT Press, 2007.

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

[11] “About Wiring,” Wiring. Accessed October 15, 2017. [Online]. Available: http://wiring.org.co/about.html

[12] *FQ777-610 Instructions Manual*, Senfa-Toys, Chenghai District, Shantou City, Guangdong Province, China, 2017.

[13] P. Cantrell, “Dissymmetry of Lift,” The helicopter aviation home page. Accessed October 1, 2017. [Online]. Available: http://www.copters.com/aero/lift_dissymetry.html

[14] “My lame IR copy toy,” dc414. Accessed October 11, 2017. [Online]. Available: https://www.dc414.org/2011/03/my-lame-ir-copy-toy/

[15] “QS5010 3.5CH Super mini infrared rc helicopter with Gyro Mode 2,” Banggood. Accessed October 06, 2017. [Online]. Available: https://us.banggood.com/Wholesale-Warehouse-QS5010-Super-Mini-Infrared-3_5CH-RC-Helicopter-With-Gyro-Mode-2-wp-Usa-941286.html?rmmds=myorder

[16] “Leap Motion in processing,” GitHub. Accessed October 10, 2017. [Online]. Available: https://github.com/nok/leap-motion-processing

[17] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, “Analysis of the accuracy and robustness of the Leap Motion Controller,” *Sensors*. May 14, 2013. [Online]. doi:10.3390/s130506380

[18] M. M. Sturman, D. E. Vaillancourt, and D. M. Corcos, “Effects of aging on the regularity of physiological tremor,” *Journal of Neurophysiology*, Jun 2005. [Online]. doi:10.1152/jn.01218.2004

[19] P. R. Hayden, “Unmanned systems: A lab-based robotic arm for grasping phase II,” M.S. thesis, Dept. of Eng. and App. Sciences, NPS, Monterey, CA, USA, 2016. [Online]. Available: https://calhoun.nps.edu/handle/10945/51716

[20] “JJRC H31 waterproof headless mode one key return 2.4G 4CH 6Axis RC Quadcopter RTF,” Geekbuying. Accessed October 06, 2017. [Online]. Available: https://www.geekbuying.com/item/JJRC-H31-Waterproof-RC-Quadcopter-368859.html

[21] Analog Devices, *AD8403 digital potentiometer*, C01092-0-2/02(C), 2002. [Online]. Available: http://biakom.com/pdf/AD8402_ad.pdf

[22] J.G. Ziegler and N.B. Nichols, “Optimum settings for automatic controllers,” *Trans. ASME*, vol. 64, pp. 759–768, Nov. 1942. [Online]. Available: http://chem.engr.utc.edu/Student-files/x2008-Fa/435-Blue/1942-paper.pdf

[23] “Ziegler-Nichols method,” Michigan Technological University. Accessed October 07, 2017. [Online]. Available: http://pages.mtu.edu/~tbco/cm416/zn.html

[24] “Prime 17W,” OptiTrack. Accessed October 26, 2017. [Online]. Available: http://optitrack.com/products/prime-17w/specs.html

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California