RC You Later: Development of a Remote-Controlled Hovercraft with First-Person View Capabilities

A thesis proposal presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honor's college.

Spring 2024

Presented by:

Jackson Wade

Thesis Committee:

Dr. Elissa Ledoux, Thesis Director

Dr. John Vile, Thesis Committee Chair

RC You Later: Development of a Remote-Controlled Hovercraft with First-Person View Capabilities

by Jackson Wade

APPROVED:

_____

Dr. Elissa Ledoux, Thesis Director
Lecturer, Department of Engineering Technology

_____

Dr. John Vile, Thesis Committee Chair
Professor of Political Science and Dean, University Honors College

Acknowledgements

I received help and support from numerous people throughout the entirety of this project.

First and foremost, I would like to thank my team members from MAJJK—Manal Malek, Adam Rice, Jacob Alexander, and Katelyn Justice. Their hard work and commitment to senior design allowed us to create a functioning RC hovercraft. They also graciously permitted me to utilize the project to further my honors thesis and donated parts for me to complete my individual progression.

I would like to thank my Thesis Director, Dr. Elissa Ledoux, next. Her guidance, support, and expertise mentored me throughout the project and thesis. The work and time she placed into both should not go unnoticed and is very appreciated. In addition, I am grateful for Dr. Hongbo Zhang as he was the voice of reason who initiated my change of plans. Without this interaction and suggestion, the RC hovercraft would not be where it is today.

Next, I would like to thank the Engineering Technology Department at Middle Tennessee State University. The department provided a project budget to each team in senior design. This financial support made the project possible.

Finally, my friends served a large role outside of the project as my support group. Through all the stress and troubles brought on by the project, they stood by my side motivating me to keep moving forward. Some friends donated parts and time to help in any way they could.

Abstract

This paper follows the journey of the user interface subsystem for a remote-controlled hovercraft with first-person view capabilities. The interface includes a controller, goggles, and micro-camera. I established the subsystem within senior design and individually progressed the subsystem with the inclusion of a head tracker to provide a new operator experience. The head tracker allows the user to maneuver the camera through natural head movements on the pitch and yaw axis. Head tracking is a relatively new technology that introduces new means of control. The application of such technology is endless, from assistive technology to gaming or to the battlefield.

The project's purpose is to apply the skills and knowledge I have learned to design and create a complex component of a mechatronic system while exploring the application of technology. The design, testing and analysis, and lessons learned are discussed to describe the growth of the project as well as myself.

Table of Contents

List of Figures

List of Tables

Introduction

A. Project Background

Over the courses Mechatronic System Design and Automation System Design, a Mechatronics Engineering student at Middle Tennessee State University (MTSU) is expected to complete a senior design project. The first course focuses on the design and completion of subsystems while the second on their integration. The base project was created while working within a team, also known as MAJJK, consisting of five members. The team decided to construct a remote-controlled (RC) hovercraft with first person-view (FPV) capabilities to provide a thrilling pilot perspective. Two user experiences were formulated by the team— a third person perspective (without the goggles) and a first-person perspective (includes goggles but no head tracker). The hovercraft is completed with a laser turret and an FPV headset with the intended purpose of a toy for ages 12 and up to enjoy.

B. Market Analysis and Considerations

The conception of the project had a sparse pre-existing market. Hovercrafts and RC toys individually are far from new, but the idea of an RC hovercraft is fresh; the initial concept for a hovercraft was developed by John Thorneycroft in 1870s, but it was not until 1955 that Christopher Cockerell created the first functioning model [1]. All while, RC toys were progressing exponentially. The first instance of an RC toy was in 1898 by Nikola Tesla and his RC torpedo in New York City [2]. Quickly followed, RC airplanes and boats hit the market in 1937 and 1940s respectively [3]. Next, RC cars were available to European consumers in 1960s, and the United States received their first

RC car models in 1970s. Finally, the first RC hovercraft was introduced in 1988 by Japan [4].

Currently, products on the market tend to possess basic functionality that are either expensive and bulky or small and cheaply manufactured. MAJJK was determined to challenge the market by focusing on a concise build and determining the least expensive components while maintaining quality. Furthermore, the team included the FPV capabilities to progress the simplistic toys on the market. The team applied research for a proof of concept regarding the usefulness and available market for an RC hovercraft. The team focused on three subsets of a market: total available market (TAM), serviceable available market (SAM), and serviceable obtainable market (SOM). TAM is the "total revenue opportunity available for a product" while SAM is what can be "realistically serve[d]" based on the companies "current product offerings" [5]. Finally, SOM refers to the "market share [of] a company" that is achievable within a specific time frame. The breakdown of each is described within **Table 1** below. The estimated cost per unit was established to be approximately $500 for the purpose of this analysis [6].

Table 1. Market Analysis

Market	Units Sold Per Year (million)	Revenue Per Year (billion)	Description
Total Available Market (TAM)	131.43	65.72	Number of households in USA in 2023 [7]
Serviceable Available Market (SAM)	64.9	32.45	USA population of people ages 10-24 [8]
Serviceable Obtainable Market (SOM)	2.07	1.035	North American remote control product market [9]

The reasonability of the market motivated the team to continue with the design process. MAJJK then determined the project goals and specifications for the project.

The goals for the project were to create an operational RC hovercraft with a laser turret and incorporated FPV interface system. The design specifications formed by the team for the entire project were as follows:

A. Vehicle should move under its own power;

B. Turret should move independent of vehicle;

C. User interface should include controls for steering, thrust, and firing projectiles;

D. Hull should be self-contained;

E. Overall weight should be less than 45 kilograms;

F. Project dimensions should fit within a volume of 1m-by-1m-by-1m;

G. Project should contain safety attributes;

H. Project should remain within a budget of $1000;

I. Battery life should last 30 minutes or more;

1. Project should move at least 2 meters per second.

C. Individual Subsystem and Progression

The project was split into subsystems; these subsystems can be defined as follows: lift, thrust, FPV interface system with controller, and laser turret. Each subsystem was assigned to a team member; I am responsible for the design and construct of the control and communication system that formulates the user experience. This includes the microcontroller, remote controller, FPV goggles, and camera. My research, designs, and prototypes are discussed in further depth in later sections.

To progress the subsystem on an individual journey, I determined the creation of a third user experience which will be first person with the inclusion of a head tracker to the FPV goggles. Originally, the user had to manually move the camera through the controller while operating the hovercraft. The implementation of a head tracker will recognize the user's head movements and mimic the movements within the camera's turret, automating the viewing experience. The natural head movements (pitch and yaw) are measured by an integrated gyroscope; these values are communicated by transceivers to the servos of the turret. The yaw value controls the rotation around the vertical axis (y-axis) while the pitch rotates about the lateral axis (x-axis). Ultimately, this will provide the user with a seemingly real piloting experience as well as an exploration of valuable technology.

Tools and Development

A. Tools and Software

For this project's duration, I relied on the standard engineering tools and resources available on campus in the Voorhies Engineering Technology Building, the Davis Science Building, and the James E. Walker Library. Each location provided free access to the necessary tools and software. These include but not limited to the following:

1. AutoCAD: A software that allows the creation and design of 2D and 3D objects. This software also forms blueprints, assembly files, and stress analysis [10].
2. 3D printer: This style of printing translates digital files into 3D objects through an additive manufacturing process by overlaying a heated filament that hardens as it cools [11].
3. Arduino IDE: This software is the development environment for Arduinos. This is where the programing for the microcontroller is written and applied based on the wiring of the Arduino [12].
4. Flowchart creator: A software that allows the user to create a graphical representation of a workflow or process. This visual aid is very common for computer algorithms and system controls.
5. Printed circuit board (PCB) design: A circuit board consisting of pre-developed copper connections. An online software is used to create the connections and set the types of layers needed for the PCB. This file is sent to a manufacturer to build. This allows for compact size, increased reliability, and quality control [13].
6. Circuit simulator: A circuit simulator provides an interactive software space for users to build and test wiring diagrams and components prior to building with the physical

parts. This lowers the chance of mistakes which could cause safety issues and unnecessary costs [14].

7. Soldering iron kit: Soldering is a technique to join two or more electrical components together utilizing heat and metal alloy, solder. A kit is comprised of the soldering iron with stand, solder, sponge, and the required protection equipment (safety goggles, helping hands, and fume extractor) [15].
8. T8S by Radiolink Android app: Application available by the controller's manufacturer to program the controller with ease [16].

B. Foundational Work and Collaboration

A portion of the information and data regarding this project originated from teamwork over Spring and Fall of 2023. My fellow members of Team MAJJK are fully aware and consented to the use of the original findings and work for me to build upon. Credit for their work will be explicitly stated in the respective areas. My personal research and work for the subsystem I was responsible for will be the primary focus of this paper.

In addition, I sought outside guidance and consultations from the professors of the Department of Engineering Technology at MTSU to aid in my research and work. I also received outside training and certifications on equipment and software mentioned previously through the university.

Methodology

Stage I: Design and Development

A. Individual Subsystem

The user interface subsystem for any RC vehicle is the integration point for all subsystems; thus, the variety of parts (goggles, controller, and microcontroller) required planning and cohesion with the other subsystems to ensure compatibility. This motivated my goals, research, and designs as the work set within the first semester set the team up for a seamless integration transition. The first semester only required that each subsystem was functional independently. For this subsystem, the FPV capabilities and beginnings of a controller were required. Note that the head tracker was initially part of the original project but dropped due to time constraints to focus on finalizing the overall project. Discussion over the research and progress will be discussed later in the system expansion.

i. Goals

My goals for this stage were to establish video communication between the goggles and camera. My stretch goal was to construct the remote controller with a simple program to lead into the next semester strong. Since integration is the focus of Stage II, it was not a primary objective to have the controller completed.

ii. Prior Work Research

The initial plan was to build the FPV goggles and convert a video game controller. This was decided on to provide a challenge while maintaining familiarity with the target audience. I relied on the information regarding FPV provided through the racing drone market and hobbyists to determine the ideal attributes.

To direct my research, I interviewed Nathan Fisk, a fellow student in the mechatronics program, who has been a RC drone and plane hobbyist for 7 years in 2023. I asked questions pertaining to his experience of building and collecting similar RC vehicles to gather a better picture of what the team needed to accomplish. His first advice was to obtain complete software and hardware harmony. He also explained that higher end RC vehicles provide longer flight times, power, smoother control, and clear video.

The first attribute I explored was the goggle shape and size to determine the most comfortable and inclusive for the user. Currently, there are two styles of goggles: box and low profile [17]. Both provide similar functionality but notable differences. The most notable difference is the size; low-profile goggles offer a smaller, compact style while box goggles tend to look bulky. Due to this, box goggles can feel heavy or even uncomfortable depending on the straps and overall weight. Furthermore, the screen type utilized within the goggles juxtaposes each other. Box goggles utilize a "special magnification lens" coupled with a normal liquid crystal display (LCD) screen while low-profile goggles implement a "small LCD screen for each eye" that require "special magnification lens to focus and enlarge" the image [18]. The importance of this difference would be the extra considerations necessary for the separate screens to ensure an optimal viewing experience. For instance, the distance measured from the center of one eye to the next, or interpupillary distance (IPD), would determine the screen placement to assure the user is perceiving a clear image. This value fluctuates by person; therefore, the team would need to determine if an average would suffice, different size goggles, or a way to adjust the screen IPD within the goggles. Next, I explored the screen itself. The aspect ratio, resolution, and field of view had to be considered for the screen.

Firstly, the aspect ratio is dependent on the camera output that is being used. The majority of FPV cameras on the market that I found used either 4:3 or 16:9. Secondly, the resolution value is the number of pixels on the screen and indicates the quality of the LCD panel. The quality is a linear relationship with the resolution value; a high-resolution value indicates better quality. With this, the cost aspect of the screen would need to be weighed since higher resolution screens are more expensive. Finally, the field of view is how big the screen appears to the user. This is calculated based on the edges of the viewing plane and the angle to the center of the eye. This range is on average within "25 degrees to around 80 degrees."

The next attribute I considered for the goggles was the video receiver and antenna. To establish communication, the receiver and transceiver must be on the same frequency band; this includes compatibility amongst the other components such as camera, antenna, and screen. Based on sources, 5.8 GHz frequency band is the "most widely used frequency for flying with FPV goggles" [19]. Yet, other attributes such as directionality, polarity, and locality of the antenna are important to consider. Antennas can either be directional or omni-directional. The difference is how the radio waves radiate; directional is more focused in a singular path while omnidirectional sends equal waves in all directions. Furthermore, polarization can either be linearly or circularly polarized. Linear is the most basic as it utilizes one plane to transmit a signal in an oscillating pattern. A circular polarization will use two planes and transmit signals in a rotating pattern. Signal using circular polarization can be lost due to a mismatch of signal spin from the transmitter to receiver. Finally, the placement of the antenna is important to keep in mind as interference can come from other radio waves or obstructions through

various materials. In summary, a box style goggle with a directional, linearly polarized antenna that all functions on the 5.8 GHz frequency band will be the most practical for this project. To pair with this, the micro-camera would also need to be on the same frequency band.

B. Early Prototypes

i. Design

Based on the research, I came up with an initial design for the goggles. The sketch is shown in **Figure 1**.

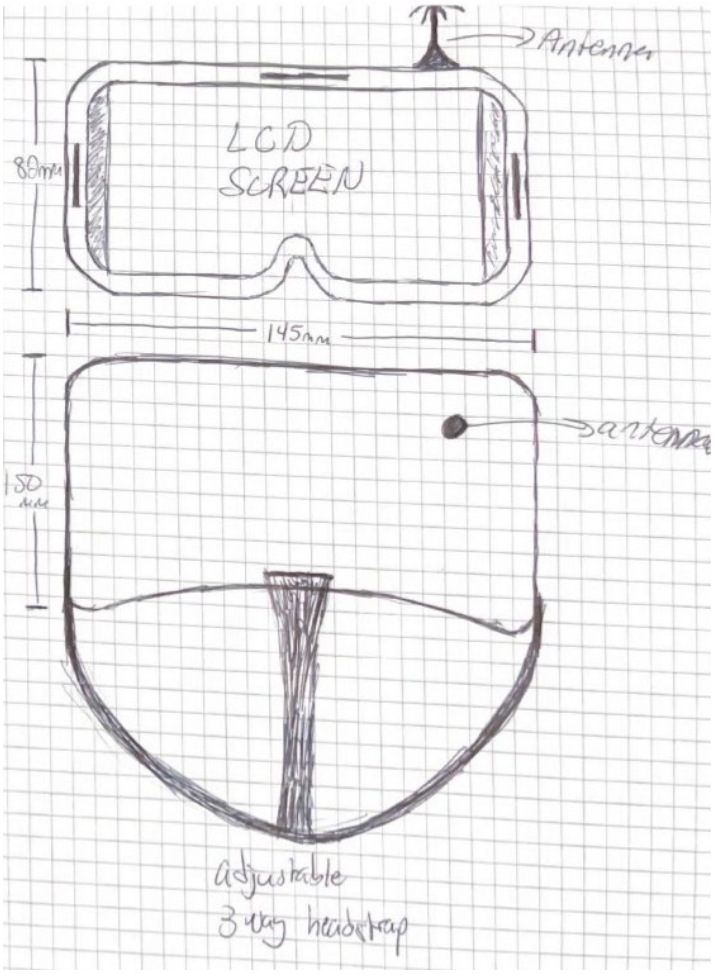

Figure 1. Initial design sketch for the FPV goggles.

I included approximate dimensions based on average goggles on the market. The design consists of the primary attributes I determined was ideal during research. It also includes a three-way adjustable head strap for an inclusive and comfortable approach for users. At this point, I had not determined a detailed design sketch as I still needed to determine the necessary parts to send the video feed from the micro-camera to the goggles. This research resulted in a cost analysis that will be discussed within the next section. Based on the analysis, I purchased pre-manufactured goggles and a micro-camera. The subsystem only required powering the two components and ensuring both were on the same channel for the communication to be set.

Next, I created a simple signal flow chart for the controller, refer to **Figure 2**. The diagram showcases the controller communication to the hovercraft. The user creates an input that is processed in the microcontroller. The signal is sent to the transmitter on the controller that will be wirelessly connected to the receiver on the hovercraft. The signal is received and interpreted by the microcontroller on the hovercraft, resulting in the desired output.

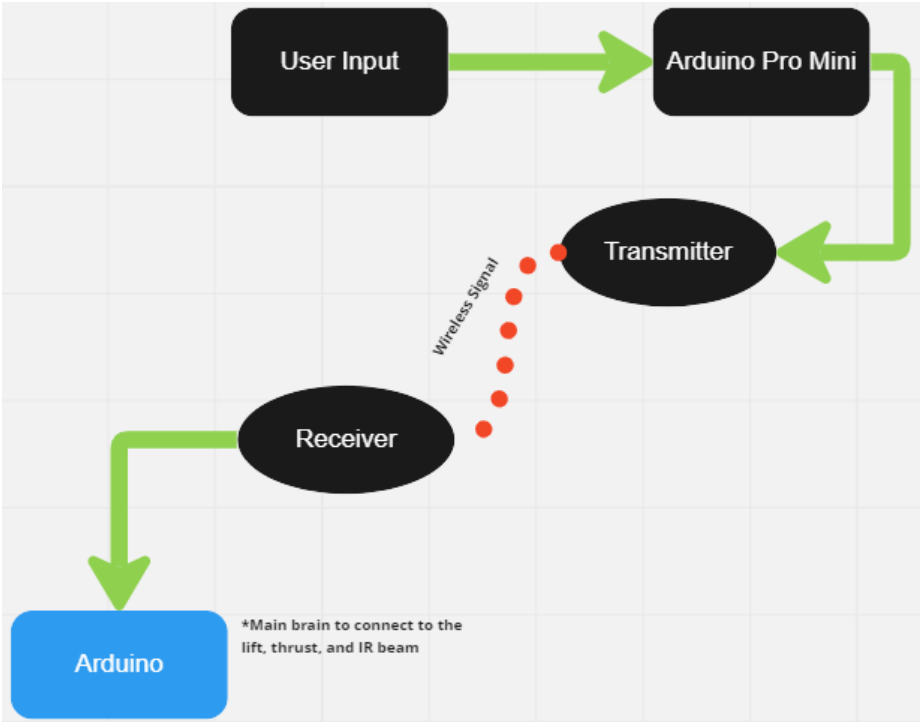

Figure 2. Signal flow diagram from controller to hovercraft.

Through research on controllers, I found a do-it-yourself (DIY) Instructable written by Dejan Nedelkovski, a mechatronics engineer who has been utilizing Arduino for almost ten years. Nedelkovski posts detailed instructions regarding projects using Arduinos that include steps, videos, materials, code, and diagrams to guide people on various projects. Specifically, he created a tutorial called "DIY Arduino RC Transmitter" that serves the purpose of a remote controller [20]. I liked the overall design and user interface; therefore, I utilized this as a guide to create the controller. The instructions included the circuit diagram in **Figure 3** below.

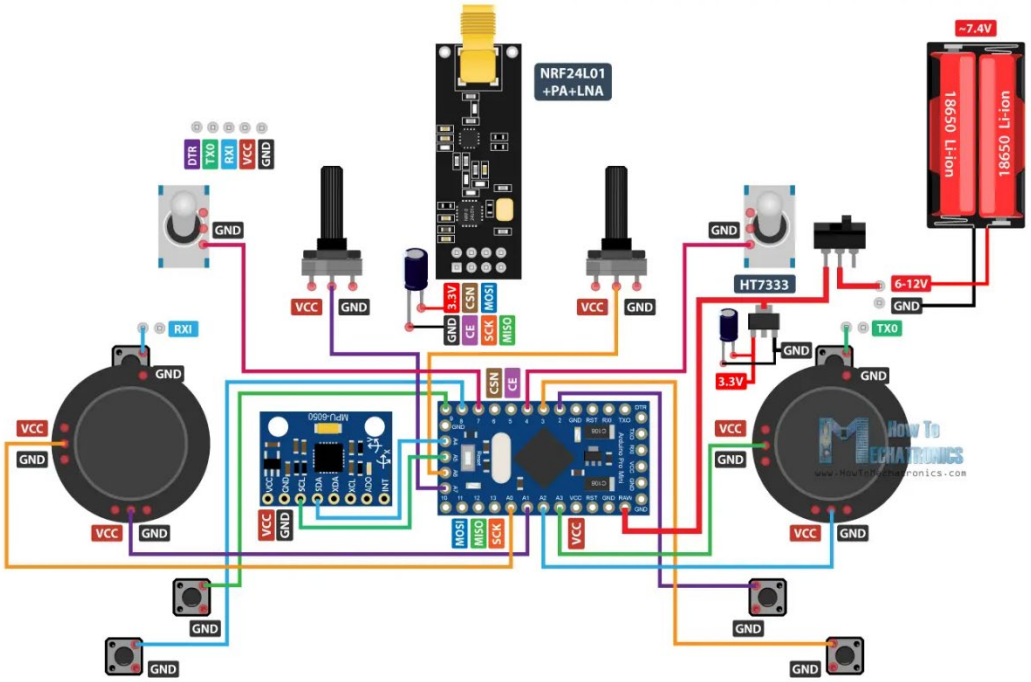

Figure 3. Circuit diagram of the controller provided on the website [20].

This diagram guided me on the PCB design. This step is not necessary for electronics, but it cleans up the wiring, condenses the size, lowers errors in wiring and assembly, and supports long-term reliability and safety in the connections. I used a free online software to recreate the PCB design using Nedelkovski's YouTube video walking through the steps. The resulting design, **Figure 4**, was sent to a manufacturer to create the actual boards.

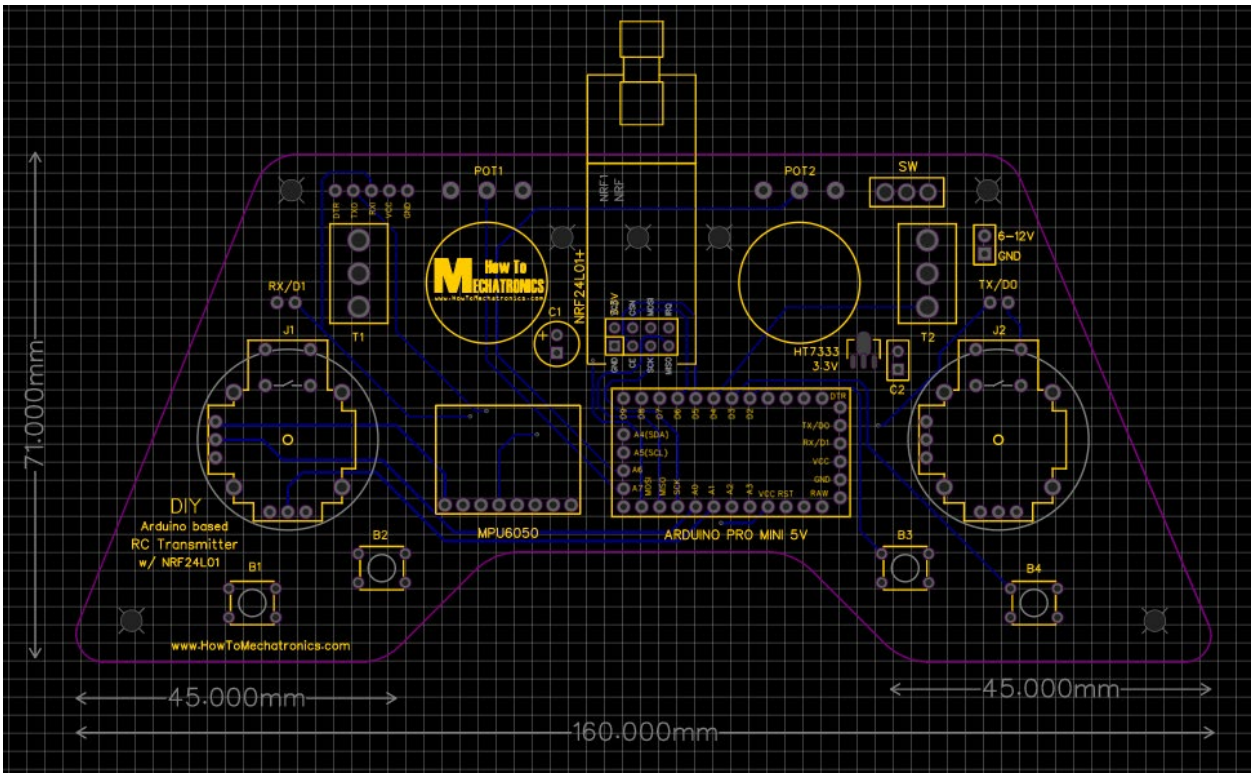

Figure 4. PCB design with the designated connections between components.

I was content with the product I received and continued to the step of soldering the components in place. The parts necessary to create the controller are listed within **Table 2**. The MPU6050 is not important for the original concepts of the project but would allow the team room for future possibilities for the controller. Thus, I decided to leave this component on the design. I also stayed with the nRF24L01 module as this component is a transceiver, which means the component can do the role of both the transmitter and receiver. Furthermore, the instructions called for 6 push buttons; however, I purchased joysticks with built-in push buttons.

Table 2. Parts List for Controller

Part	Quantity Used
Joysticks (Xbox 360)	2
3.3V Voltage Regulator	1
Toggle Switch Slide	1
NRF24L01+PA+LNA	1
PCB	1
Toggle Switch	2
Potentiometer	2
Mini USB to TTL Connector Adapter Module	1
Sparkfun pro mini-AT mega board	1
3.7V 18650 Lithium Battery	2
Lithium Battery Holder for 2	1
Push Buttons	6*
MPU6050*	1
L-shaped Pin Headers	11
Straight Pin Headers	6
100µF Capacitor	2

ii. Testing and Analysis

As I searched for parts to build the FPV subsystem (goggles and camera), I noticed that the expense was considerable; thus, I decided to conduct a cost analysis to determine if it was worth building the goggles or purchasing manufactured goggles. The results are listed within **Table 3** below. Note that the cost analysis does not include pricing on creating the enclosure of the DIY goggles as the plan was to 3-D print this aspect. Furthermore, extra costs for wiring, aesthetics, production, and shipping are not considered. As the table shows, the costs to build the subsystem is approximately $52.91 more than the expense of purchasing the pre-manufactured option. Hence, I determined the ideal route was to purchase what was available on the market.

Table 3. FPV Goggles and Camera Cost Analysis

DIY		Pre-Manufactured	
Part	**Cost ($)**	**Part**	**Cost ($)**
5 in LCD Screen	31.08	Goggles	53.53
11.1V 2200mAh LiPo Battery	12.99		
RC305 - 5.8Ghz 8 Channel AV Receiver	16.94		
Fresnel Lens 3X Magnification	4.99		
Adjustable Elastic Head Strap	8.05		
WR832 5.8GHz 40CH Receiver	26.40	5.8GHz Camera with Transmitter	20.84
5.8GHz Camera with Transmitter	20.84		
3 Way DIP Switch	5.99		
3.7V 1S LiPo Battery	14.13	3.7V 1S LiPo Battery	14.13
Total	**141.41**	**Total**	**88.50**

Finally, I was not able to collect any data or analysis of the controller. I was adjusting the wiring of the controller when the positive prong of the battery touched the negative pin header on the back of the controller. This caused the microcontroller to spark and damage the board electrically, making the board unusable. This incident occurred at the end of the semester and left no time to recreate for the end of the semester demonstration. Thus, the controller was delayed till the following semester. However, the soldered controller can be observed within **Figure 5** and the function of each component is described in **Table 4**.

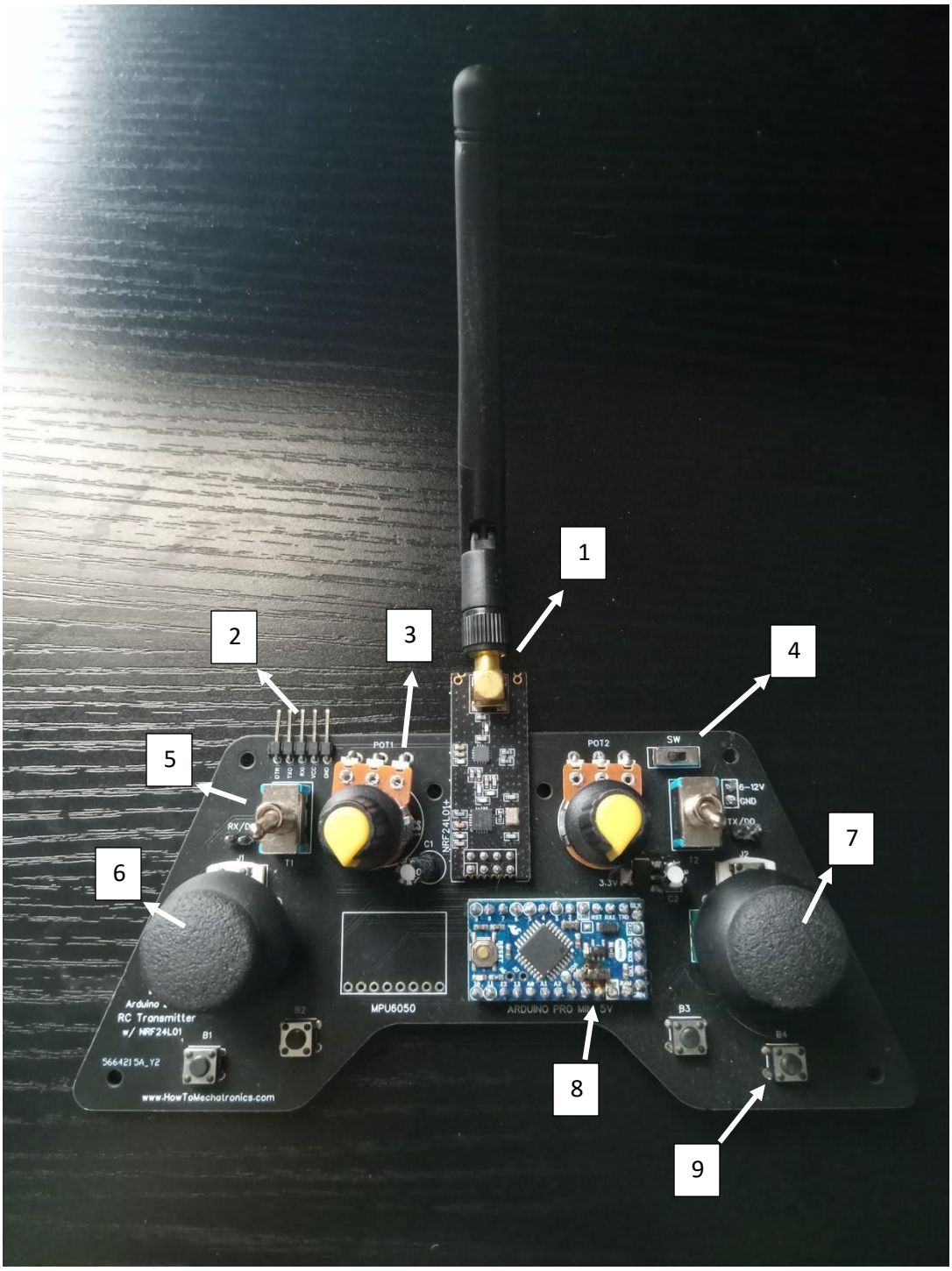

Figure 5. Diagram of controller marking major parts used for user interface.

Table 4. Controller Parts and Purpose

Reference #	Component Name	Purpose
1	NRF24L01	Transmits signal of user inputs
2	Programming Header Port	Connects Arduino Pro Mini to a Mini USB to TTL Connector Adapter Module to load controller's program
3	Potentiometer	Adjusts lift fan
4	Slide Switch	Turns controller on and off
5	Toggle Switch	Actuates emergency switch and/or safety lock
6	Left Joystick	Adjusts thrust fan
7	Right Joystick	Turns servo left and right
8	Arduino Pro Mini	Routes signals as a microprocessor
9	Push Button	Actuates laser

iii. Final Product of ENGR 4580

The final prototypes for the first semester were presented to the class and group of professors. The FPV subsystem, **Figure 6**, was working by the end of the semester, meeting the requirements of the course. At this point, I did not have the 1S LiPo battery, so I utilized the Arduino's 3.3V output as a power source for the micro-camera at first. I was experiencing issues with the camera where it would shut off randomly or not turn on at times. I consulted with a member of the group who had years of electrical experience. I learned my issues were due to insignificant current despite being in the voltage range. I resolved the issue by switching to the breadboard power supply module provided in the Arduino kit given to the group by the department. This allowed me to use a 9V 1A power

supply AC/DC adapter that plugs into a wall socket. The power supply module contains voltage regulators to drop the output voltage to either 3.3V or 5V depending on the channel used. This option provided the necessary voltage based on the camera specifications and sufficient current flow to maintain the video feed. The circuit for the camera is shown in the bottom right of **Figure 6**.

Figure 6. FPV subsystem portrayed working.

The goggles and camera that I selected for the project provide the following specifications:

Goggles:

- LCD Screen: 3.0in, 480*320-point high brightness
- Display Ratio: 16:9
- Receiver: 5.8GHz, 40ch

- Battery: 3.7V/1200mAh built-in, rechargeable
- Size: 5.31in-by-5.19in-by-2.55in

Camera:

- Weight: 4.5g
- Battery: 3.2-5V, 1S LiPo
- Transmitter: 5.8 GHz frequency, 40 channels
- Size: 0.77in-by-0.6in

These meet the goals set at the beginning while maintaining quality and cost.

C. Lessons Learned

Over the semester, I learned a lot regarding the design process and applicating the topics learned thus far. I learned the importance of applying the technology that is already available on the market instead of reinventing the wheel. This saves time and money. I learned that relying on others for their experience and strengths is a great way to maneuver around obstacles. It also creates a strong team and productive work environment as people flourish in their strengths while others learn. Next, I learned complacency creates room for disasters. I ignored safety when moving wires around with the controller and it resulted in a set-back for the project. Furthermore, the incident could have caused injury. This emphasized the safety protocols taught within courses of maintaining wire awareness and shutting all power while manipulating wires.

Stage II: Integration, Testing, and Refinement

A. Design Progression

The purpose of Automation System Design is to integrate all subsystems to create a complete product. Within this semester, MAJJK improved on the previous prototype by creating an enclosed hovercraft body powered by one power source. The user interface portion of my subsystem provided a bridge from the user input to a desired reaction. My primary focus overall was establishing communication as this is the heart of integration; therefore, most of my research and work was directed to this topic.

1.1. Goals

My goals at this stage were to first establish a stable communication between the controller to hovercraft. Next, I needed to create a signal channel for each input to the corresponding output. This will allow the user to control each subsystem. Finally, I wanted to clean up the final design and program to make the project presentable and user friendly.

1.2. Research

To understand how to establish communication, I needed to learn more regarding the nRF24L01 transceiver module, as I did not have prior experience with this component. I first examined the datasheet and pinout of the module. I took note that the "module operates at 3.3V" over a distance of "100 meters" on the "frequency of 2.4GHz" [21]. Since the modules are not made in the United States, it is important to consider regulations for wireless communications as certain frequencies and channels are set aside for federal use. According to the Federal Communications Commission (FCC), 2.4GHz is legal in all countries; however, only channels one through twelve can be used [22].

Furthermore, the module is compatible with Arduino with an appropriate range for an RC toy. Next, I learned that the module utilizes serial peripheral interface (SPI) communications to transmit signals. Arduino has pre-existing libraries created to establish this communication and interface at ease. Finally, I recorded the pinout configuration and purpose for the module. **Figure 7** shows the pinout and **Table 5** describes the function for each pin. Both were provided through the datasheet [21].

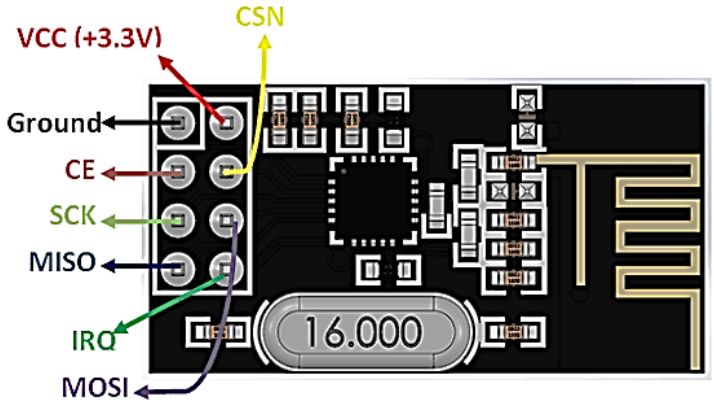

Figure 7. Pinout of nRF24L01.

Table 5. Pinout Configuration of nRF24L01

Abbreviation	Pin Name	Purpose
Ground	Ground	Connects to system's ground
Vcc	Power	Powers the module
CE	Chip Enable	Enables SPI communication
CSN	Chip Select Not	Disables SPI unless value is HIGH
SCK	Serial Clock	Provides clock pulse for SPI
MOSI	Master Out, Slave In	Connects to receive data from microprocessor
MISO	Master In, Slave Out	Connects to send data from microprocessor
IRQ	Interrupt	Remains low and is used only if interrupt is required

It was also noted within the reference that the modules tend to be tricky due to "many cloned versions in the market." Regarding these issues, it was suggested to add a capacitor in parallel to the power and ground on the module, inspect the power supply value, and check for interference.

Next, I searched the internet for other projects that used the nRF24L01 module with Arduino to establish communication. This provided circuit diagrams and sample codes to practice establishing communication, as well as to prototype test before applying the controller's code. The tutorial of the controller I built included the code; however, it is easier to start with a small program to troubleshoot any issues. I was able to locate two simple tutorials. The first tutorial is from the previous creator, Nedelkovski, that focuses on communication of the nRF24L01 module [23]. I planned to recreate the "Hello

World" example first before moving to the other tutorial I found. The other tutorial establishes an LED that is controlled by a push button [24].

B. Final Prototype

i. Design

The "Hello World" circuit was provided within the instructions and is depicted within **Figure 8** below.

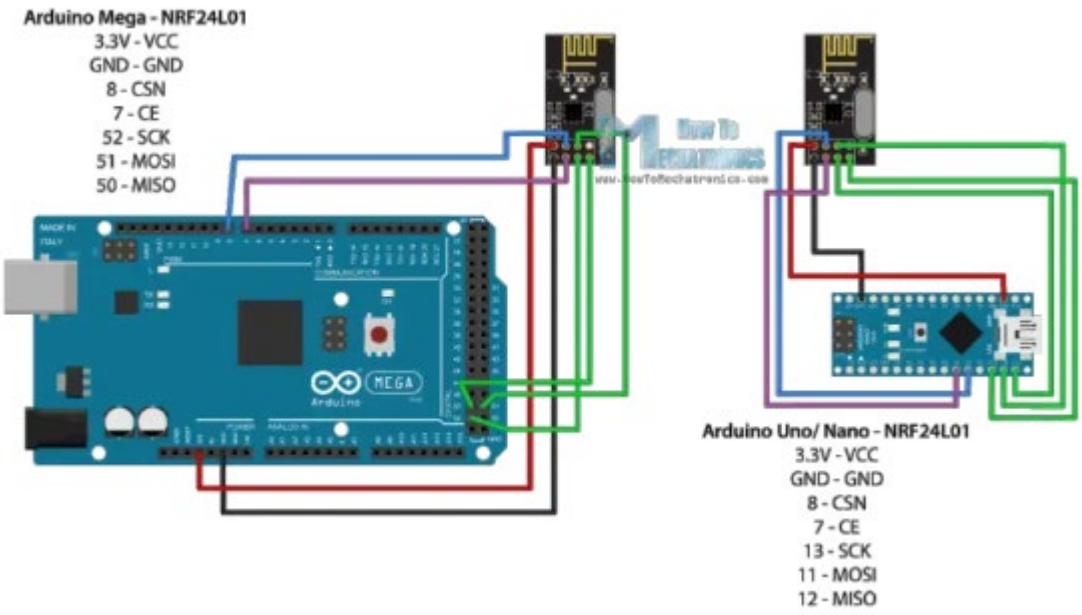

Figure 8. Circuit of both the transmitter and receiver to establish communication [23].

To run the code, I was required to download the RF24 library on the Arduino IDE to utilize pre-written functions. Since the nRF24L01 module is a transceiver, either microprocessor can play the role of transmitter or receiver. The code applied to the microprocessor determines the role the module plays either by telling the system to write (transmitter) or read (receiver). The code for both roles is provided in **Figure 9**.

Transmitter Code

```
/*
* Arduino Wireless Communication Tutorial
*     Example 1 - Transmitter Code
*
* by Dejan Nedelkovski, www.HowToMechatronics.com
*
* Library: TMRh20/RF24, https://github.com/tmrh20/RF24/
*/

#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

void setup() {
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}

void loop() {
  const char text[] = "Hello World";
  radio.write(&text, sizeof(text));
  delay(1000);
}
```

Receiver Code

```
/*
* Arduino Wireless Communication Tutorial
*       Example 1 - Receiver Code
*
* by Dejan Nedelkovski, www.HowToMechatronics.com
*
* Library: TMRh20/RF24, https://github.com/tmrh20/RF24/
*/

#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
}

void loop() {
  if (radio.available()) {
    char text[32] = "";
    radio.read(&text, sizeof(text));
    Serial.println(text);
  }
}
```

Figure 9. Provided code for the roles of transmitter (left) and receiver (right) [23].

Both codes start by including the libraries used. Failure to include the libraries will provide errors when calling specific functions such as the 'RF24 radio(7,8).' This function requires two arguments, the CE and CSN assigned ports, to create an RF24 object. Next, each code creates a byte array to act as a specific address, 00001. These match as the goal is for the transmitter to talk to the specific receiver. Next, the code establishes the identity and means of each nRF24L01 module. To begin, the type of communication is determined. The transmitter is set to write while the receiver is to read while using the specified address. The following line is required to set the power amplifier level. This level is based on the distance between the transmitter and the

receiver. Finally, the role of the module is determined. The ‘radio.stopListening()’ instructs the module to be the transceiver while the receiver is determined by the ‘radio.startListening()’ function. Then, the code outlines a loop section. The transmitter assigns the string “Hello World” into an array of characters and sends the message to the receiver. The receiver outputs the message to the serial monitor. The transmitter delays for one second and runs this section of code again, creating an output on the receiver side every second. It is important that the declared ports for CE and CSN match with the physical circuit of the Arduinos. In this case, the ports used are seven and eight. Furthermore, it is important to adjust the serial monitor’s band rate to match the value within the code, 9600. This step is vital because it will display unrecognizable characters instead of the desired output.

The next tutorial provided the beginnings of the hovercraft’s code. The circuit diagram used in the tutorial is shown within **Figure 10**.

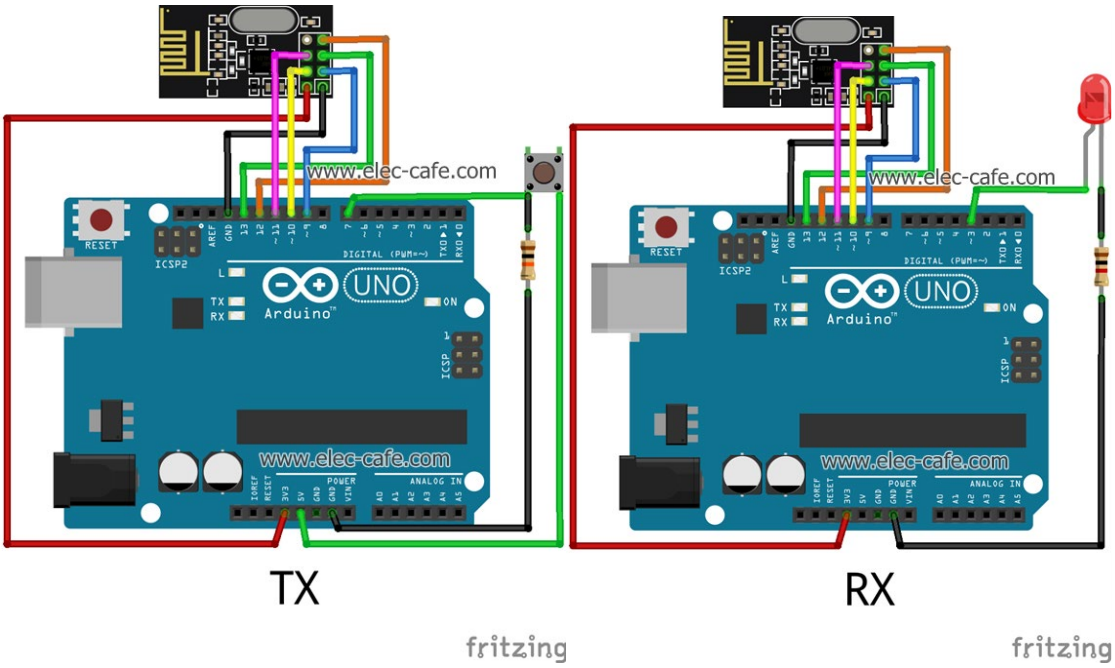

Figure 10. Circuit diagrams of the transmitter and receiver for a push button-controlled LED [24].

Just as before, I applied the code to the microprocessor of corresponding roles. The code for the transmitter is displayed in **Figure 11**.

Upload Code to TX

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
int msg[1];
RF24 radio(9,10);
const uint64_t pipe = 0xE8E8F0F0E1LL;
int SW1 = 7;

void setup(void){
Serial.begin(9600);
radio.begin();
radio.openWritingPipe(pipe);}

void loop(void){
if (digitalRead(SW1) == HIGH){
msg[0] = 111;
radio.write(msg, 1);}}
```

Figure 11. Transmitter code that sends message when button is pressed.

This code is very similar to the previous tutorial; however, it does vary in the defined address used to communicate and the message being sent. There is no difference between the types of address; it is only required to be consistent between the transmitter and receiver. In addition, it is important to assign inputs and outputs of the Arduino within the definitions of the code. This helps to adjust based on the wiring or a quick reference to how the circuit should be wired. In this case, the push button is assigned to pin seven of the Arduino. Finally, the loop section checks the state of the button: HIGH or LOW. In the case the state is HIGH, the message is sent to the receiver at the specific address. If the button state is LOW, no message is sent.

Now, the receiver code is portrayed in **Figure 12**. Within this code, the LED is assigned to pin three of the Arduino. Since a user input is controlling the LED, it is important to include the “pinMode(Pin Number, Mode)” function within the setup

portion. The two arguments for the function require the pin number that the LED is assigned to and the desired designation. Since the pin number was saved to the variable 'LED1' and the desired action is output, the way it will be written in the code is "pinMode(LED1, OUTPUT)." Finally, the loop section consists of a Boolean flag. While the flag is not true, the code will loop; otherwise, the error message of 'no available radio' will display. The actual loop saves the message into an array that is checked. If the message matches, then the LED is instructed to switch to HIGH state (on). If the message does not match, then the LED is set to LOW state (off).

Upload Code to RX

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
int msg[1];
RF24 radio(9,10);
const uint64_t pipe = 0xE8E8F0F0E1LL;
int LED1 = 3;

void setup(void){
Serial.begin(9600);
radio.begin();
radio.openReadingPipe(1,pipe);
radio.startListening();
pinMode(LED1, OUTPUT);}

void loop(void){
if (radio.available()){
bool done = false;
while (!done){
done = radio.read(msg, 1);
Serial.println(msg[0]);
if (msg[0] == 111){delay(10);digitalWrite(LED1, HIGH);}
else {digitalWrite(LED1, LOW);}
delay(10);}}
else{Serial.println("No radio available");}}
```

Figure 12. Receiver code that is connected to the LED [24].

ii. Testing and Analysis

At first, I had major struggles setting up communication between two nRF24L01 modules. The issues brought me back to my research in that there are counterfeit modules on the market now. The counterfeit, cheap modules have different specifications to work properly; this caused issues as the boards became easy to damage unknowingly. To counter this, I applied a capacitor as advised and ensured all possible causes of interference were kept at a distance. I tried this with multiple modules, all with the same outcome of not communicating. Thus, I further researched these issues and learned that nRF24L01 modules are prone to manufacturing defects. Batches of the modules have been known to come with damaged antennas and poor PCBs. A few hobbyists recommended buying more than one batch to increase the chances of obtaining working modules. I deemed this worth an attempt as the modules is very inexpensive. This is why I ended up purchasing a few batches from various vendors to obtain real nRF24L01 modules. I was able to locate two modules that produced the output shown in **Figure 13**.

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

Figure 13. Output from the "Hello World" tutorial.

This result was only obtained once. I concluded the modules were unreliable and unstable. My attempts to troubleshoot the modules only severely delayed my progress on the project. Hence, I sought guidance regarding the modules and my available options to

determine the best path to proceed. I reached out to Dr. Hongbo Zhang, professor of a Controls Optimization class, who has experience with transmitters and receivers. During the conversation, the issues of the module were reiterated, but Dr. Zhang's concern was the project's approaching deadline. He advised me to consider other means of establishing a controller to ensure that the team had a functioning project. Dr. Zhang suggested some controllers with pre-paired receivers that would meet the needs of the project. I agreed with his observations and decided to adjust due to the time constraints.

iii. Final Product of ENGR 4590

The FPV camera and goggles did not receive any major changes from the previous semester. With the addition of the 1s LiPo battery, the camera was able to power on in a portable fashion. The simplistic wiring to connect the two together allowed the turret to move freely without damaging or tangling the camera's wires. A photo showcasing the micro-camera integrated with the turret subsystem is shown within **Figure 14**.

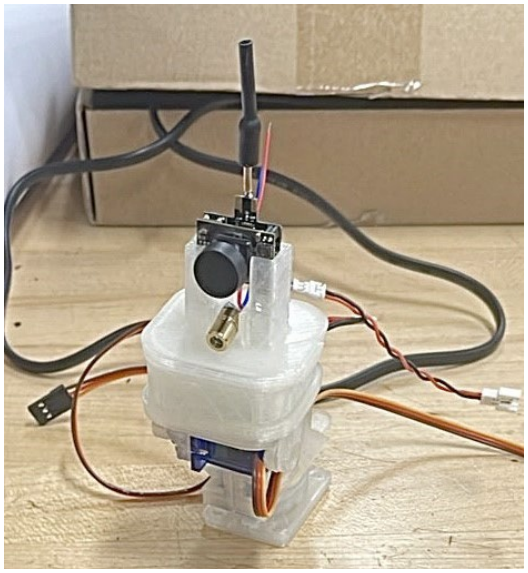

Figure 14. Turret with micro-camera.

The design of the turret includes a compartment into which the camera's battery can easily slide. This guaranteed a compact and functional design for the integration of these subsystems.

Next, the controller and pre-bound receiver I purchased was the RadioLink T8S eight channel remote controller with the R8EF receiver, which is shown in **Figure 15**.

Figure 15. Purchased controller (left) and paired receiver (right).

I decided this controller was sufficient for the purpose of the project as it provided enough output channels, user familiarity of the shape, and ease of adjusting the controller and program. The joysticks on the controller can be modified. For instance, the controller in **Figure 15** has centered joysticks that will automatically spring back to the center. I deemed this style to be unintuitive for the user as I planned to utilize the left joystick for the thrust. By removing the spring, the gimbal will remain in any position without constant user input. This attribute would also allow the home position to be set in the down position, so the stick's progression forward could correlate to the speed of the motor.

The rest of the changes I made to the controller were through programming. The manufacturer of the controller includes a user-friendly application for adjusting the inputs

and outputs of the controller. A screenshot of the application while being connected to the controller is shown in **Figure 16**.

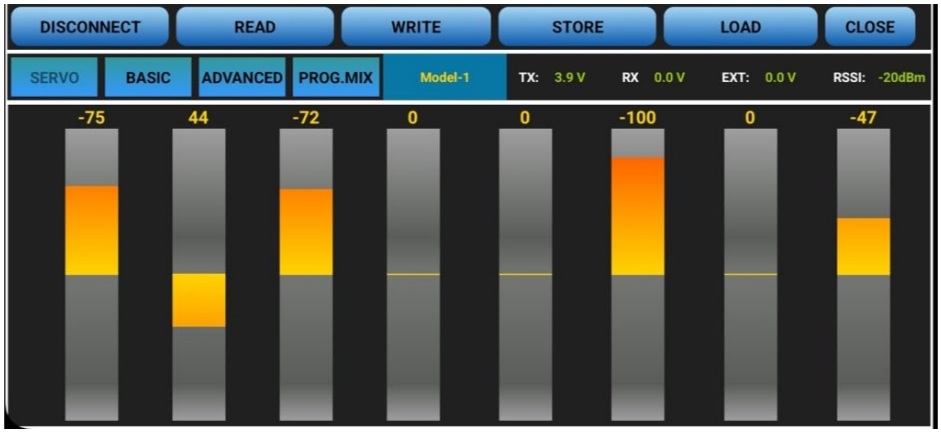

Figure 16. Android user interface to program controller showing the eight channels.

The application depicts the signal of each channel, one to eight from left to right, as the input is manipulated. Based on the input, I assigned the necessary channels to roles. I made my decisions on what would be the most user-friendly based on video games. **Table 6** outlines each channel assignment and its purpose.

Table 6. Remote Control Channel Assignments

Channel	Input	Purpose
1	Right Joystick (Left & Right)	Turret x-axis
2	Right Joystick (Up & Down)	Turret y-axis
3	Left Joystick (Up & Down)	Thrust Fan Speed
4	Left Joystick (Left & Right)	Servo Swivel
5	Three-way Switch	N/A
6	Intermittent Button	N/A
7	Three-way Switch	N/A
8	Scroll Wheel	Lift Fan Speed

Since the hovercraft included electronic speed controlled (ESC) motors, I was able to use the component's regulated 5V supply to power the receiver without additional batteries to

the system. Hence, I was able to connect all the output components of the hovercraft to the designated channel ports to make the hovercraft work.

Upon establishing a functioning RC hovercraft, it is vital to ensure safety measures are in place. In the RC world, the term is “fail-safe” and means a technique implemented to prevent rogue RC vehicles when communication is lost between the transmitter and receiver. This can occur by exceeding the transmission distance, turning off the controller before the receiver, low battery, or interference. The default for each channel was set to 50, meaning that each channel will be set to an input of 50 upon losing communication. This caused issues only within the motors as the default value resulted in both fans being set to half power with no means of control. To make this safer, I adjusted these values to zero to power off the motors of the hovercraft until communication can be re-established.

C. Lessons Learned

Through finishing the project with the team, I learned valuable lessons and sharpened important skills. I learned the importance of spending time in the research phase; specifically, I learned the value of being specific and thorough in the research. Team MAJJK realized the majority of the obstacles we ran into already had existing technology and resources on how to apply said technology; yet these solutions were often discovered after the team changed course. In addition, I learned that mistakes are inevitable, and plans will change during any learning and professional journey. These failures do not mean that one is not capable, but rather that a different path exists. I had to learn to give myself grace after failing. Finally, I was able to strengthen my

troubleshooting, research, and teamwork skills while overcoming the failures presented in the project.

Stage III: System Expansion

A. Subsystem Progression

I deemed establishing a wireless head tracker to my subsystem would be an appropriate progression due to a desire to conquer the pending challenges remaining after completion of the senior design project. I was determined to master this application of technology to showcase the skills and knowledge I have learned in my journey as a mechatronics student at MTSU.

i. Goals

My goals for this stage were to design and develop a head tracker that manipulates two servos utilizing the pan and tilt axis to mimic the user's head movements while operating the hovercraft. The signal flow chart and rendering of the head tracker are shown in **Figures 17-18**.

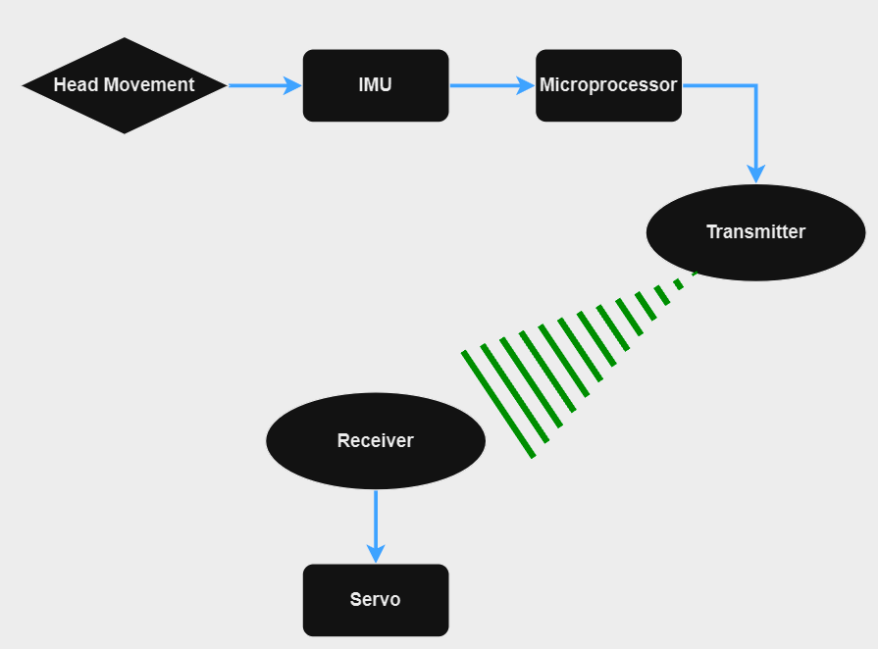

Figure 17. Signal flow chart for the head tracker.

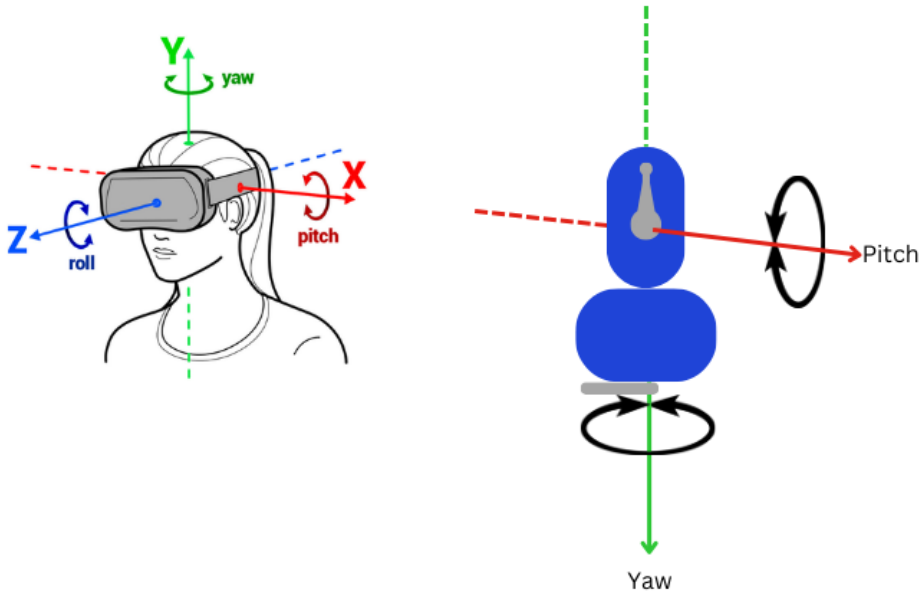

Figure 18. Drawing depicting the mimicked movements of the head tracker to the two servos.

The head tracker must be a stand-alone component for those who wish to use the FPV goggles with no head tracker. The design should also maintain or add to the user experience; this includes comfort, usability, and accuracy.

ii. Research

Based on prior knowledge, an accelerometer and gyroscope were established as practical sensors for a head tracker. An accelerometer measures the acceleration of an object while a gyroscope measures the orientation of an object relative to the original frame. Both sensors combined would be able to mimic the direction of movement and the speed of that movement; however, both sensors are "prone to errors" from issues such as "noise and drift" that causes the "data uncertainty" to increase with every measurement [25]. There are a variety of ways to eliminate this issue, such as applying a filter equation

or including a magnetometer, which measures the gravitational force to limit the drift. I determined it would be easier to apply a nine degree of freedom inertial measurement unit (IMU), which combines the three sensors onto one board. In researching various IMU chips offered on the market, I located two— BNO085 and GY-85— that could accomplish the task. At first, the team voted for the BNO085 chip; however, the pre-existing coding library was limited. This required extra coding to apply the input values to a servo, which exceeded the coding capabilities of the team. Therefore, I switched to the GY-85 chip. This chip is older and offers more resources and coding libraries, making it easier to implement.

The original plan was to use the same nRF24L01 transceiver modules from the DIY controller for the head tracker; yet the instability of establishing communication as well as the limited signal distance due to interference produced concerns. It would not be ideal for the controller and head tracker signal distances to vary significantly as this would limit the user's viewability when operating the RC vehicle. Thus, I considered having the head tracker outputs sent through the controller to the hovercraft. This means there would only be one path of communication for all inputs and one signal distance for fail-safe.

For the head tracker to transmit using the controller, I needed to establish a way for the controller to accept the head tracker as inputs. The RadioLink T8S controller does not have this capability; upon research, I learned controllers with a trainer port allow for external analog inputs. Yet, the specific kind of controller was determined by a head tracker project I found [26]. The project includes the hardware and software utilized with instructions to recreate or an option to purchase a pre-built kit [27]. Due to time

constraints and obtaining proper wires, I opted to purchase the Universal Head Tracker from the Medlin Project. The project also provides suggested controllers that are known to work. This is why I decided to purchase the FrSky Taranis X9D Plus. A photo of the controller is depicted in **Figure 19**. I decided to purchase a pre-owned controller, as this was cheaper.

Figure 19. FrSky Taranis X9D Plus used controller purchased.

The upgrade in the controller required a new receiver. I referred to the controller's manual to locate a compatible receiver [28]. I determined the FrSky X8R receiver to fit the requirements of the project, view **Figure 20**.

Figure 20. FrSky X8R receiver [29].

B. Prototype

i. Design

To start the head tracker, I determined that breaking the subsystem into parts would make the build easier. The stages were to display the input values on the serial monitor, create a functional wired head tracker, and progress with implementation of wireless communication.

I utilized an online tutorial that connects the GY-85 to an Arduino to read the values as the IMU is manipulated and outputs them on the serial monitor. The circuit diagram to connect the GY-85 is shown in **Figure 21**.

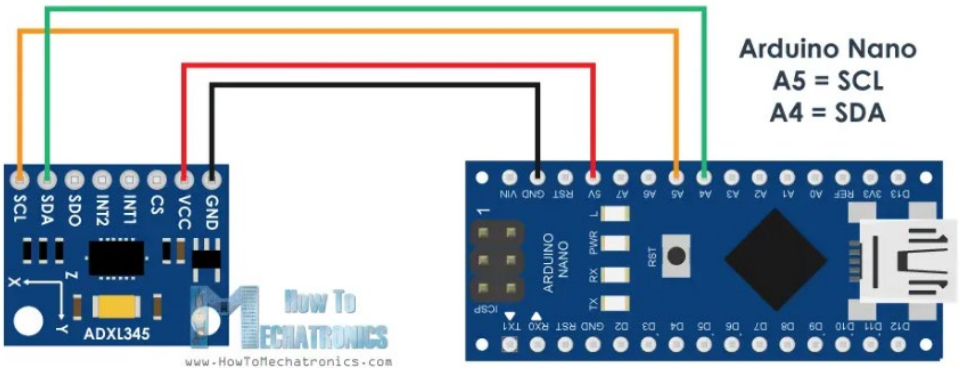

Figure 21. Pinout to connect GY-85 IMU chip to Arduino Nano [30].

Furthermore, the code was provided in the tutorial as well, and can be referenced in **Figure 22**.

```
#include <Wire.h>  // Wire library - used for I2C communication

int ADXL345 = 0x53; // The ADXL345 sensor I2C address

float X_out, Y_out, Z_out;  // Outputs

void setup() {
  Serial.begin(9600); // Initiate serial communication for printing the results on the Serial monitor
  Wire.begin(); // Initiate the Wire library
  // Set ADXL345 in measuring mode
  Wire.beginTransmission(ADXL345); // Start communicating with the device
  Wire.write(0x2D); // Access/ talk to POWER_CTL Register - 0x2D
  // Enable measurement
  Wire.write(8); // (8dec -> 0000 1000 binary) Bit D3 High for measuring enable
  Wire.endTransmission();
  delay(10);
}

void loop() {
  // === Read acceleromter data === //
  Wire.beginTransmission(ADXL345);
  Wire.write(0x32); // Start with register 0x32 (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(ADXL345, 6, true); // Read 6 registers total, each axis value is stored in 2 registers
  X_out = ( Wire.read()| Wire.read() << 8); // X-axis value
  X_out = X_out/256; //For a range of +-2g, we need to divide the raw values by 256, according to the datasheet
  Y_out = ( Wire.read()| Wire.read() << 8); // Y-axis value
  Y_out = Y_out/256;
  Z_out = ( Wire.read()| Wire.read() << 8); // Z-axis value
  Z_out = Z_out/256;

  Serial.print("Xa= ");
  Serial.print(X_out);
  Serial.print("   Ya= ");
  Serial.print(Y_out);
  Serial.print("   Za= ");
  Serial.println(Z_out);
}
```

Figure 22. Code applied to read values from the IMU [30].

The outputs were seemingly correct until further inspection. Thus, I started troubleshooting the common issues. I rechecked the circuit diagram to confirm the wiring, which allowed me to notice excessive heat being released by the IMU. This led me to check the voltage entering and exiting the IMU to ensure the voltage was within specifications. By using a multimeter, I noted the voltage was approximately 4.6V entering and leaving the IMU. This value would be within the range given on the tutorial I was viewing; however, the tutorial is written for the accelerometer, AXDL345, and not the GY-85 chip. These chips have a similar pinout and code, but different voltage specifications. The GY-85 has a lower voltage range, meaning I was causing a short-circuit in the board to the point that it was not reading values correctly. Thus, I changed the circuit to connect the power to the Arduino's 3.3V power source instead of the 5V. This resulted in the following output in **Figure 23**.

```
22:28:15.939 -> Xa= 0.69   Ya= 0.44   Za= 0.17
22:28:15.971 -> Xa= 0.71   Ya= 0.36   Za= 0.07
22:28:16.003 -> Xa= 0.41   Ya= 0.25   Za= -0.31
22:28:16.035 -> Xa= 0.51   Ya= 0.16   Za= -0.37
22:28:16.067 -> Xa= 0.71   Ya= 0.14   Za= -0.40
22:28:16.099 -> Xa= 0.75   Ya= 0.11   Za= -0.54
22:28:16.131 -> Xa= 0.83   Ya= -0.00   Za= -0.73
22:28:16.163 -> Xa= 0.93   Ya= 0.04   Za= -0.73
22:28:16.195 -> Xa= 1.07   Ya= -0.00   Za= -0.86
22:28:16.228 -> Xa= 1.03   Ya= 0.06   Za= -0.86
22:28:16.261 -> Xa= 1.07   Ya= -0.01   Za= -0.23
22:28:16.324 -> Xa= 0.54   Ya= -0.01   Za= -0.41
22:28:16.356 -> Xa= 0.82   Ya= -0.10   Za= 0.65
22:28:16.388 -> Xa= -0.07   Ya= 0.10   Za= -0.21
22:28:16.420 -> Xa= -0.28   Ya= 0.04   Za= 1.31
22:28:16.452 -> Xa= -0.61   Ya= 0.06   Za= 0.77
22:28:16.484 -> Xa= -0.36   Ya= 0.05   Za= 0.85
22:28:16.516 -> Xa= -0.04   Ya= -0.01   Za= 1.59
```

Figure 23. Output from the GY-85 IMU.

This output is as expected according to the code. Therefore, I proceeded to the next step in applying the readings to servos through a serial connection.

By applying personal experience with Arduino's, I adjusted the code and circuit for the values to be applied as outputs to the servos. I transferred the previous circuit to an Arduino Mega as MAJJK determined this microprocessor would provide enough ports for the project. The completed circuit is displayed within **Figure 24**.

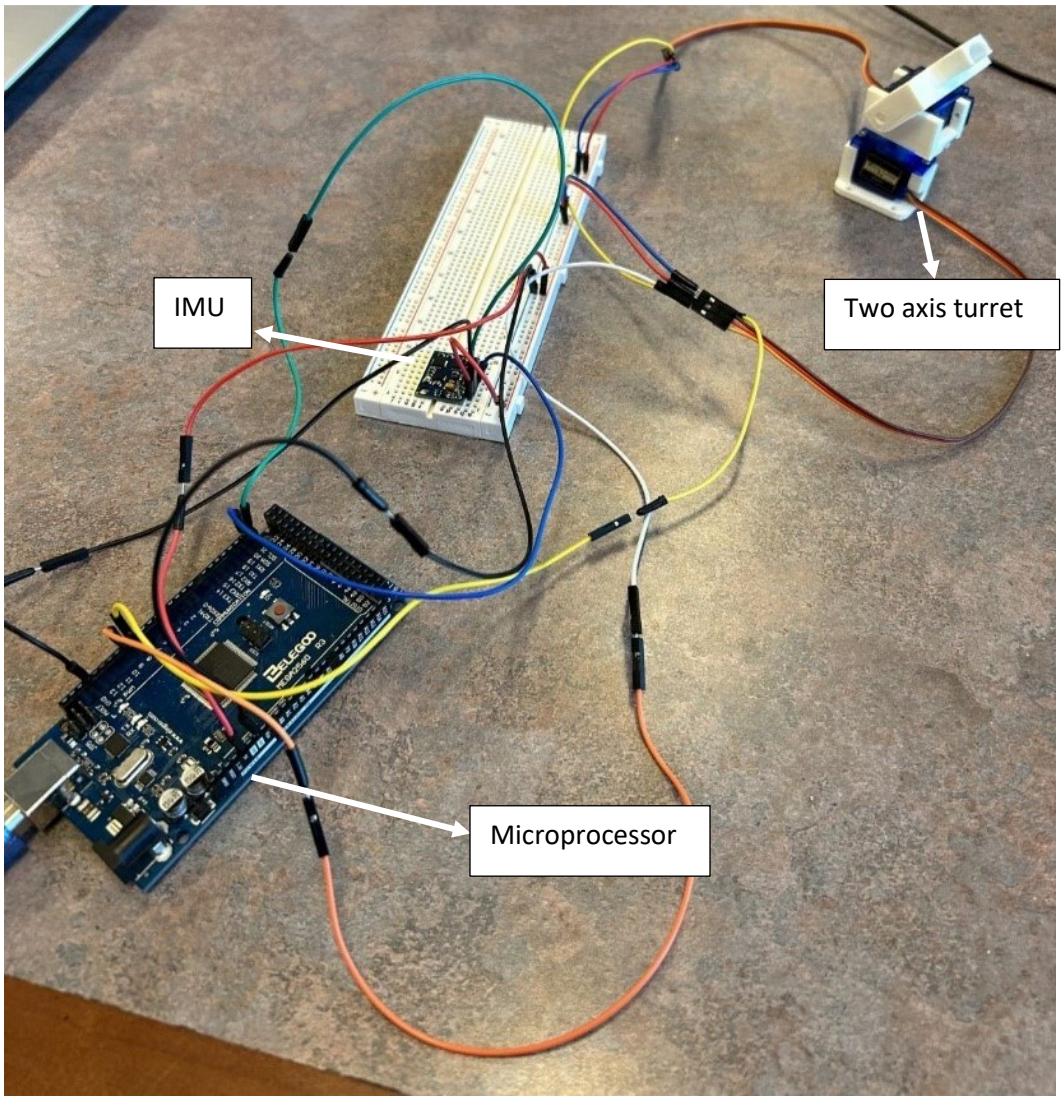

Figure 24. Wired head tracker circuit.

The code I created and applied to the microprocessor is shown below in **Figures 25-27**.

```
// Jackson Wade
// Senior Design: MAJJK
// Description: Headtracker for turret serial connection
// Used keywords file in the libraries for accelerometer on GY-85 (AXDL345) and servo
// ADXL345 Library Link: https://github.com/Seeed-Studio/Accelerometer_ADXL345/blob/master/keywords.txt
// Servo Library Link: https://docs.arduino.cc/learn/electronics/servo-motors


// Declare libraries
#include "Wire.h"
#include "ADXL345.h"
#include <Servo.h>

// Declare global variables
const float alpha = 0.5;
int fXg = 0, fYg = 0, fZg = 0;

// Name servos
Servo servoRoll;
Servo servoPitch;

// Name Accelerometer
ADXL345 acc;

```

Figure 25. Part 1 of 3 of the code for the wired head tracker.

The first portion of the code only contains the initial declaration of libraries and variables. This allowed me to utilize pre-built functions from the libraries, which removed most of the extra coding. It is important to note that I only utilized the accelerometer in this code to maintain a simplistic code until furthering the head tracker.

```
25  void setup()
26  {
27
28  // Pin assignments for servos
29  servoRoll.attach(9);
30  servoPitch.attach(8);
31
32  // Start Accelerometer
33  acc.powerOn();
34
35  // Buad rate for serial monitor to print values of roll and pitch
36  Serial.begin(9600);
37
38  // Timing declared before starting loop
39  delay(100);
40
41  }
42
43  void loop()
44
45  {
46
47  // Declare variables for math
48  int pitch, roll, Xg, Yg, Zg;
49
```

Figure 26. Part 2 of 3 of the code for the wired head tracker.

The second portion of the code contains the set-up. This includes the servo assignments to their designated ports of the Arduino Mega and powering on the accelerometer. This portion is also where the serial monitor has a baud rate assigned since the output will be displayed. The bottom of **Figure 26** shows the beginning of the loop with the declared variables. The loop is continued into **Figure 28** where the calculations to obtain the roll and pitch values as well as the commands to display are shown. The calculations are based on orientational kinematics [31].

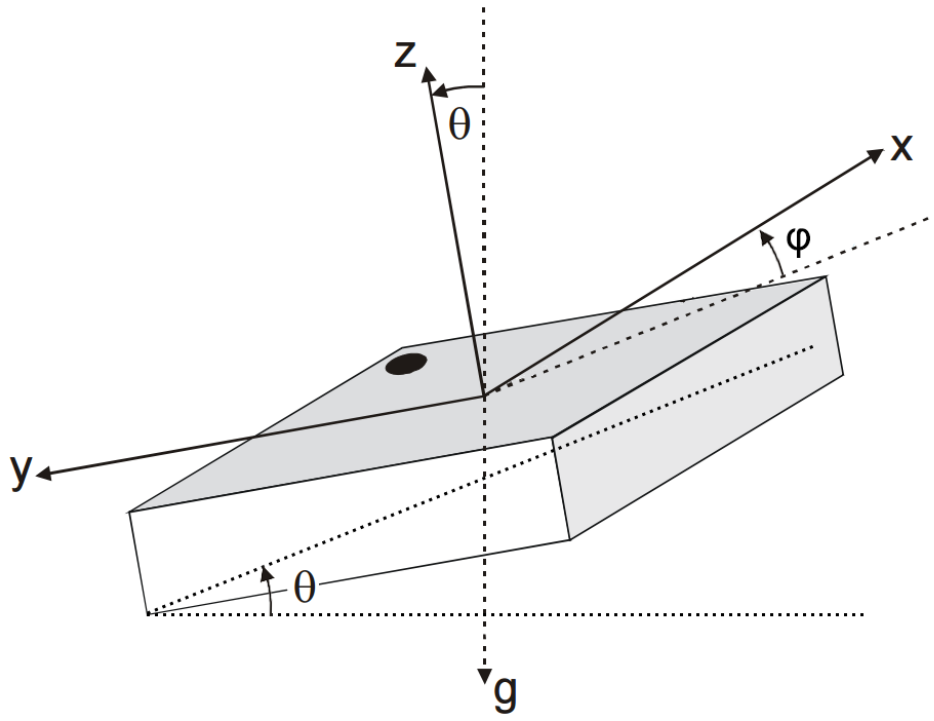

Figure 27. Diagram showing an example Cartesian and Polar coordinate relationships [32].

The relationship between the two is established using Euler angles and an orientation matrix.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & -\sin\theta \\ \sin\theta\sin\varphi & \cos\varphi & \cos\theta\sin\varphi \\ \sin\theta\cos\varphi & -\sin\varphi & \cos\theta\cos\varphi \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

Where the following variables are defined as such:

$$a_x = acceleration\ in\ the\ x\ axis$$

$$a_y = acceleration\ in\ the\ y\ axis$$

$$a_z = acceleration\ in\ the\ z\ axis$$

$$g = gravity\ constant$$

Using matrix multiplication, the acceleration in each axis can be derived to be as follows:

$$a_x = g \sin\theta$$

$$a_y = -g \sin\varphi \cos\theta$$

$$a_z = -g \cos\varphi \cos\theta$$

Yet, the roll and pitch of an IMU should be taken in reference to gravity as this serves as a constant reference direction. This provided the following equations which were used within the code.

$$Pitch = \tan\theta = \frac{a_{xg}}{\sqrt{a_{yg}^2 + G_{zg}^2}}$$

$$Roll = \tan\varphi = \frac{-a_{yg}}{a_{zg}}$$

The variables are as follows:

$$a_{xg} = acceleration\ in\ the\ x\ axis\ in\ reference\ to\ gravity$$

$$a_{yg} = acceleration\ in\ the\ y\ axis\ in\ reference\ to\ gravity$$

$$a_{zg} = acceleration\ in\ the\ z\ axis\ in\ reference\ to\ gravity$$

```
50  // Read values from accelerometer
51  acc.readXYZ (&Xg, &Yg, &Zg);
52
53  // accelerations X, Y, Z calculations based on readings; math is on scratch paper
54  fXg = Xg * alpha + (fXg * (1.0 - alpha));
55  fYg = Yg * alpha + (fYg * (1.0 - alpha));
56  fZg = Zg * alpha + (fZg * (1.0 - alpha));
57
58  // calculate roll and pitch based on acceleration values
59  roll = (atan2 (-fYg, fZg) * 180.0) / M_PI;
60  pitch = (atan2 (fXg, sqrt (fYg * fYg + fZg * fZg)) * 180.0) / M_PI;
61
62  // print of serial monitor and servo writings for movement
63  Serial.print ("Pitch: ");
64  Serial.print (pitch);
65  Serial.print (" ");
66  Serial.print ("Roll: ");
67  Serial.println (roll);
68  servoRoll.write (180 - (roll * 10));
69  servoPitch.write (180 - (pitch * 10));
70
71  // time delay for readings
72  delay (50);
73  }
```

Figure 28. Part 3 of 3 of the code for the wired head tracker.

After troubleshooting the issues that will be discussed in the next section, I was able to connect the head tracker (refer to **Figure 29)**. The head tracker requires a 3.5mm auxiliary cable and 2S LiPo battery with barrel connector.

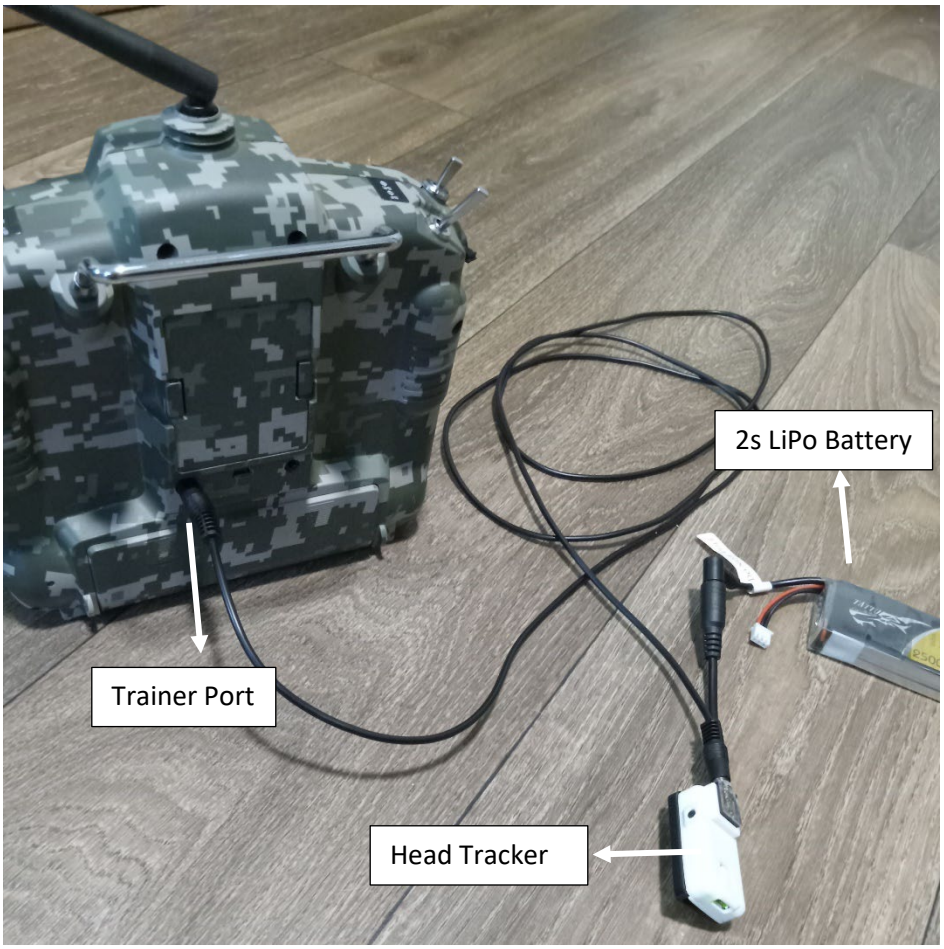

Figure 29. Photo of the head tracker connected to the controller.

I began programming the controller by defining a source as an input and assigning such to a channel on the receiver. It is important that within the model set-up of the controller to change the trainer mode to "Master/Jack;" this will allow the external inputs from the head tracker to be accepted and assigned to a channel. Then, the component of the desired output was connected to the corresponding channel on the receiver, see **Figure 30**.

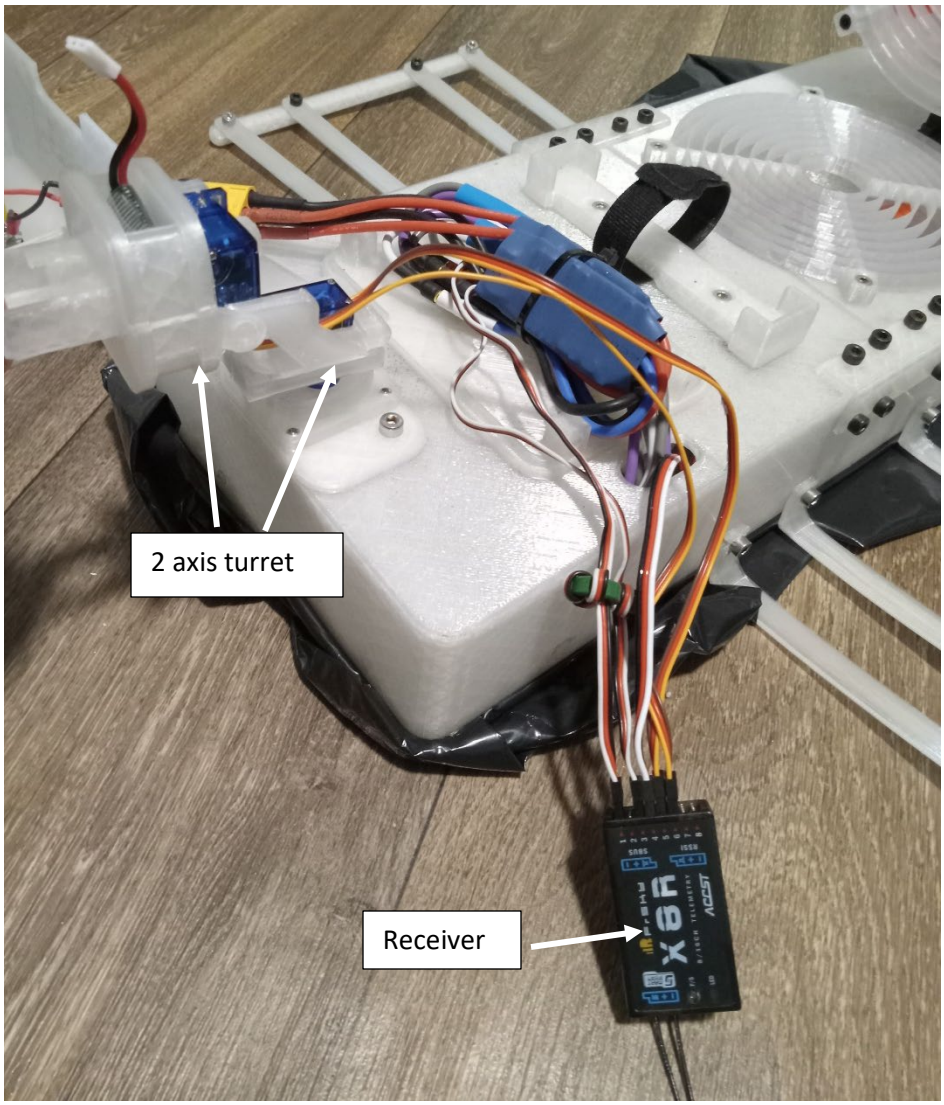

Figure 30. Photo of the hovercraft components connected to the receiver.

The assignments I made for the inputs and outputs can be referred to in **Table 7**.

Table 7. Remote Control Channel Assignments

Controller		Receiver
Input Source	**Assigned Channel**	**Purpose**
Left Stick Up and Down	1	Lift Fan Speed
Right Stick Up and Down	3	Thrust Fan Speed
Right Stick Left and Right	4	Servo Swivel
Trainer Port 6	5	Head Tracker Yaw
Trainer Port 7	6	Head Tracker Pitch

ii. Testing and Analysis

Since the controller was used, there were numerous issues from the previous owner. Firstly, the LiPo battery was poorly kept and refused to charge. Thus, I had to purchase a new battery to power the controller. After changing the battery, the controller would initiate the loading screen only to create a buzzing sound and shutting down. This would repeat until turning the controller off. Based on research, I discovered that a secure digital, SD, card update was known to be corrupt and causing issues like mine. I confirmed this by removing the SD card and powering the controller. The controller can function as normal without the SD card, minus sounds and extra firmware that is saved on the SD card. Due to this fact, I intended to leave the SD card alone; however, this plan did not last long as the controller and receiver would not bind together. I went through forums of hobbyists who had comparable results to mine to determine a solution. I ascertained that the receiver I purchased new had a newer version of the firmware than

the one on the older controller I had purchased [33]. Consequently, I needed to fix and update the SD card to flash the matching firmware to my controller.

To start, I discovered a tutorial using the OpenTX companion to flash updates for the controller and SD card [34]. OpenTX is a project that provides open-source firmware for RC radio transmitters [35]. The controller I purchased was already using this firmware. Using the companion application, I was able to check and download the latest OpenTX firmware and update the SD card contents to the latest. I tested the controller to ensure that the SD card was no longer corrupted with the new firmware. Satisfied, I moved on to adding the new firmware that matches the receiver. This firmware is provided through the manufacturer [36]. This addition required me to connect the controller to my laptop with a USB-A to Mini USB data transfer cable. Through this connection, I was allowed access to open the contents of the folders like I would with a flash drive. Using instructions I found, I simply moved the downloaded firmware into the "FIRMWARE" folder on the SD card [37]. I was now able to access this using the controller and instruct the controller to flash the firmware on the internal module. Finally, I reattempted to bind the controller with the receiver and was successful.

I connected to the hovercraft as mentioned previously. One of the first adjustments I had to make was with the thrust fan. The thrust fan is connected to the right joystick, a center position gimbal, which was sending a signal to the fan to run at half speed. By changing the offset, I made the center position a value of zero. Next, I adjusted the offset of the thrust servo for the home position to be perpendicular to the hovercraft's forward direction. Finally, I began assessing the head tracker. I began with the limitations of the head tracker to cooperate with the turret subsystem built by two members of

MAJJK over senior design. The turret system was originally believed to have extreme design limitations, but I uncovered a simple orientation change would alleviate this issue, which required an adjustment to one of the 3-D printed parts. Therefore, I planned the limitations of the head tracker on the new orientation. I manipulated the offset to zero the servo based on the desired orientation, and I increased the weight of the source to include a wider rotational range.

iii. Final Product

The final product after the progression of my subsystem had a few adjustments from senior design. The camera mount for the turret needed to be rotated 90 degrees upwards to sit in the correct orientation for the turret design. The new orientation and camera mount can be seen within **Figure 31**.

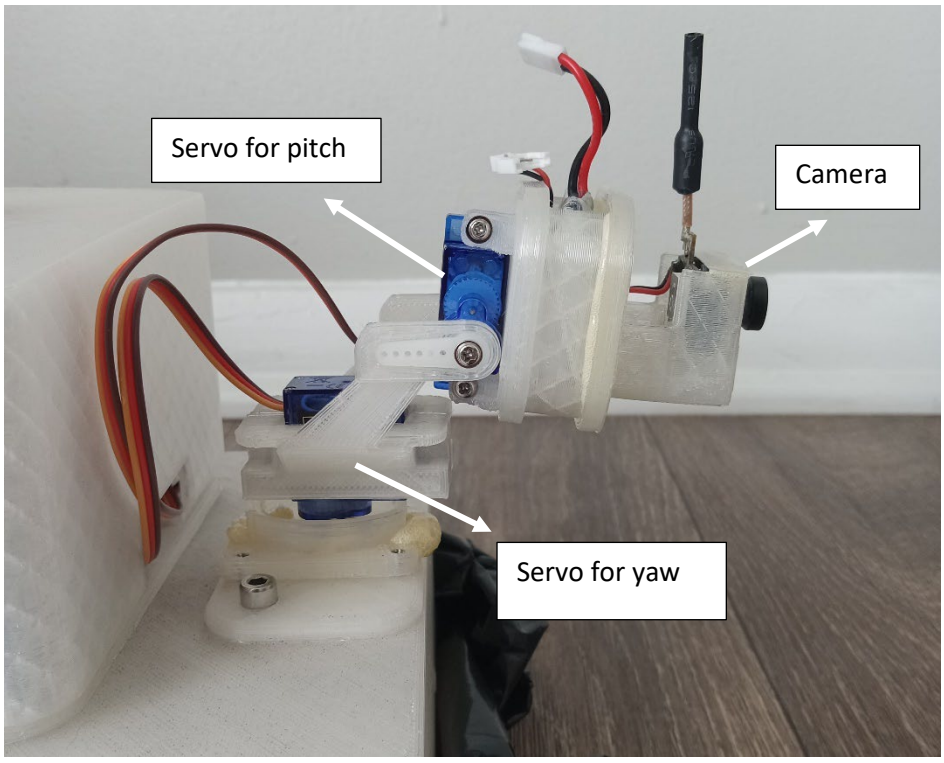

Figure 31. New turret orientation with new camera mount.

Next, I attached the head tracker to the head strap of the goggles with a 3-D printed clip. I also attached the LiPo battery for the head tracker to the top of the goggles using command strips. The placement of the head tracker was determined based on user comfort while wearing the goggles. The full user interface can be seen in **Figures 32 – 33**.

Figure 32. Full user interface system for FPV hovercraft.

Figure 33. Close view of the head tracker mounted on goggles (left) and the full system being used (right).

Finally, the completed hovercraft is viewable in **Figures 34 – 35**.

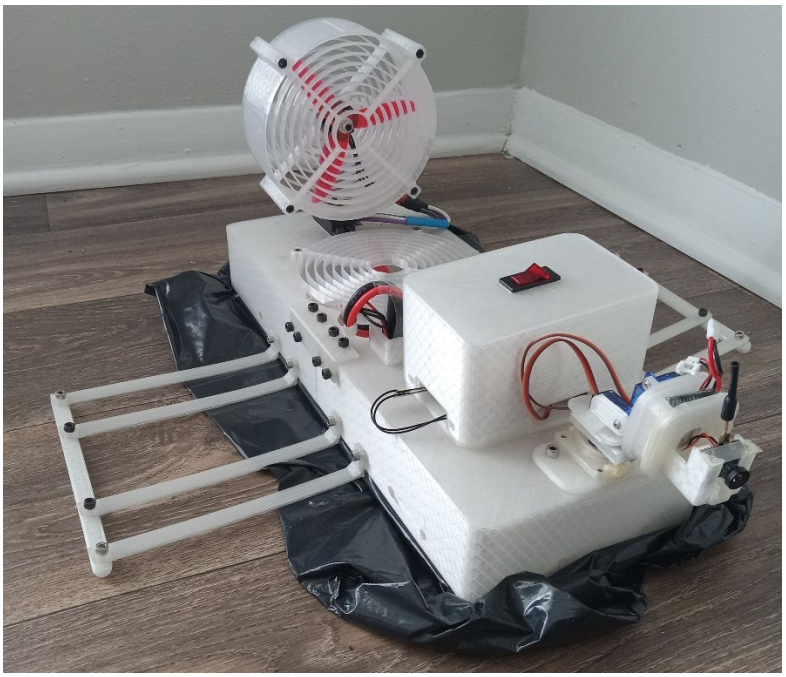

Figure 34. Side view of the final hovercraft.

Figure 35. Top view of the final hovercraft.

C. Lessons Learned

Through the progression of the head tracker, I learned a significant amount regarding the technology. I was able to apply each facet of mechatronics to establish the head tracking technology with the FPV subsystem. Additionally, this project provided me the opportunity to acknowledge my exponential growth as an engineering student. I found confidence in my abilities to troubleshoot and problem solve. Furthermore, I attained the power of dedication and perseverance.

Final Remarks

Head tracking is a new, vital control technology that can advance automation in various subject matters. I had the opportunity to explore this technology through the progression of my subsystem on the RC hovercraft built in senior design. Over the course of this project, I was able to establish a functional user interface consisting of a controller, micro camera, goggles, and head tracker. The project was long, but it proved to be indispensable for my learning experience. I have gained new skills and knowledge from team members and professors; furthermore, I have strengthened my engineering education through application in a team and individual setting.

Definition of Terms

Aspect Ratio

1. the ratio of the width of a television or motion-picture image to its height

Field of View

1. the angular extent of the observable world that is seen at any given moment

Firmware

1. computer programs contained permanently in a hardware device

First-Person View

1. of, relating to, or occurring in the first person
2. experienced as if through the eyes of an avatar

Head Tracker

1. enables an application to recognize and identify a user's head movements

Interface System

1. the place at which independent and often unrelated systems meet and act on or communicate with each other
2. region or space separating two systems, applications, or components that interact by exchanging information

Lithium-Polymer (LiPo) Battery

1. a rechargeable battery that uses lithium ions as the primary component of its electrolyte

Liquid Crystal Display (LCD) Screen

1. an electronic display that consists of segments of a liquid crystal whose reflectivity varies according to the voltage applied to them

Microcontroller

1. a microprocessor that holds memory and associated circuits
2. controls some or all functions of an electronic device or system (similar to a "brain")

Pitch

1. the action or a manner of pitching
   especially **:** an up-and-down movement

Receiver

1. a device for converting signals (such as electromagnetic waves) into audio or visual form: such as
   a) **:** a radio receiver with a tuner and amplifier on one chassis

Remote-Controlled

1. controlled (as by a radio signal) from a distance **:** operated by means of remote control

Resolution

1. a measure of the sharpness or fineness of an image

2. usually expressed as the total number or density of pixels in the image

Servomotor or Servo

1. a rotary actuator or linear actuator
2. allows for precise control of angular or linear position, velocity, and acceleration in a mechanical system

Subsystems

1. a system that is part of a larger system

Third person-view/perspective

1. outside point of view
2. birds eye view

Transceiver

1. An electronic device that is utilized within communications that can both transmit and receive radio waves

Transmitter

1. a set of equipment used to generate and transmit electromagnetic waves carrying messages or signals

Yaw

1. the action of yawing
   especially **:** a side-to-side movement

References

[1] Er. L. Talwar, "Design and Fabrication of a Remote Controlled Hovercraft,"*InternationalJournal of Innovative Research in Science, Engineering and Technology*, vol. 7, no. 3, pp. 1111–1117, Mar. 2018. doi:10.15680/IJIRSET.2018.0703131

[2] S. Beschloss, "Object of interest: Remote Control," Object of Interest: Remote Control, https://www.newyorker.com/tech/annals-of-technology/object-of-interest-remote-control

[3] T. Adam, "The history of the R/C toy [infographic]," TTPM, https://ttpm.com/the-history-of-the-rc-toy-infographic/#:~:text=1898%20%E2%80%93%20Nicolas%20Tesla%20features%20a%20remote%20control,torpedo%20is%20still%20considered%20the%20first%20R%2FC%20boat

[4] "1988 Taiyo Typhoon Hovercraft (Global) - Review, valuation, history," Tyco Taiyo Collectors, https://tycocollectors.com/product/1988-taiyo-typhoon-hovercraft-global/

[5] C. Meabe, "Tam, Sam, som: How to calculate them for your industry," Foundation, https://foundationinc.co/lab/tam-sam-som

[6] E. Ciobanu, "Guide for buying your first racing drones," Droneblog, https://www.droneblog.com/guide-for-buying-your-first-racing-

drones/#:~:text=How%20much%20does%20a%20drone,range%20from%20%24300%20to%20%24500

[7] V. Korhonen, "Number of households in the U.S. from 1960 to 2023," Statista, https://www.statista.com/statistics/183635/number-of-households-in-the-us/

[8] N. Research, "United States population by age - 2023 united states age demographics," Neilsberg, https://www.neilsberg.com/insights/united-states-population-by-age/

[9] A. Kumar and I. Dash, "Remote Control Products - hobby market size & share, by product type (drones, airplanes, cars, trains); distribution channel type (offline: Hypermarkets, convenience stores; online: E-Commerce, company websites) - Global Supply & Demand Analysis, growth forecasts, statistics report 2024-2036," Research Nester, https://www.researchnester.com/reports/remote-control-products-hobby-market/2771#:~:text=Remote%20Control%20Products%20-%20Hobby%20Market%20size%20is,of%203%25%20during%20the%20forecast%20period%2C%20i.e.%2C%202024-2036

[10] K. Ahmad, "What Is AutoCAD and What Is It Used For?," MUO, Apr. 23, 2022. https://www.makeuseof.com/what-is-autocad/

[11] 3DPrinting, "What is 3D printing? How does a 3D printer work? Learn 3D printing," 3D Printing, 2012. https://3dprinting.com/what-is-3d-printing/

[12] Arduino, "What is Arduino? | Arduino Documentation," *docs.arduino.cc*, Mar. 09, 2023. https://docs.arduino.cc/learn/starting-guide/whats-arduino

[13] Z. Peterson, “What is a Printed Circuit Board? Make Circuits by Connecting Components,” *Altium*, Oct. 05, 2020. https://resources.altium.com/p/what-is-a-pcb

[14] “IOT Wiring Diagram - Find us for search of the schematics, wiring diagrams and technical photos.” https://www.176iot.com/

[15] admin, “What is a Soldering Iron? (All You Need to Know),” Engineering World, Nov. 24, 2021. https://www.engineeringworldchannel.com/soldering-iron/

[16] “RadioLink T8S(BT) User Manual,” ManualsLib, https://www.manualslib.com/manual/2233936/Radiolink-T8s-Bt.html

[17] R. Hargrave, “FPV Goggles: The Ultimate Guide (2024 update),” RChobby Lab, https://rchobbylab.com/fpv-goggles/#How_to_Choose_the_Best_FPV_Goggles_for_You

[18] Alex, “Everything you need to know about buying FPV goggles,” Unmanned Tech Blog, https://blog.unmanned.tech/everything-you-need-to-know-about-buying-fpv-goggles/#:~:text=Other%20things%20to%20consider%201%20HDMI%20input%20This,Faceplate%20Fan%20

[19] Admin, “Best FPV goggles,” Camera Enthusiast, https://camera-enthusiast.com/best-fpv-goggles

[20] D. Nedelkovski, “DIY Arduino RC Transmitter,” How To Mechatronics, https://howtomechatronics.com/projects/diy-arduino-rc-transmitter/

[21] “NRF24L01 Wireless RF module,” Components101, https://components101.com/wireless/nrf24l01-pinout-features-datasheet

[22] G. Hardesty, “Legal & illegal frequencies & channels in the United States,” Data-Alliance, https://www.data-alliance.net/blog/legal-illegal-frequencies/

[23] D. Nedelkovski, “NRF24L01 – how it works, Arduino Interface, circuits, codes,” How To Mechatronics, https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/

[24] “NRF24L01+ basic control led (Arduino),” ELEC-Cafe, https://www.elec-cafe.com/nrf24l01-basic-control-led-arduino/

[25] B. Baker, “Apply sensor fusion to accelerometers and gyroscopes,” DigiKey, https://www.digikey.com/en/articles/apply-sensor-fusion-to-accelerometers-and-gyroscopes

[26] “Head tracker V2.1: Head tracker v2.1,” Head Tracker v2.2, https://headtracker.gitbook.io/head-tracker

[27] “Universal head tracker,” Medlin Drone Store, https://medlin-drone-store.myshopify.com/products/pre-built-head-tracker

[28] FrSky 2.4GHz ACCST Taranis X9D Plus Manual, https://www.frsky-rc.com/wp-content/uploads/2017/07/Manual/X9D%20Plus.pdf

[29] “X8R - FRSKY - lets you set the limits,” FrSky, https://www.frsky-rc.com/product/x8r/

[30] D. Nedelkovski, “How to track orientation with Arduino and ADXL345 accelerometer,” How To Mechatronics, https://howtomechatronics.com/tutorials/arduino/how-to-track-orientation-with-arduino-and-adxl345-accelerometer

[31] M. Pedley, “Tilt sensing using a three-axis accelerometer,” Freescale Semiconductor, https://www.nxp.com/docs/en/application-note/AN3461.pdf

[32] “How_to_Use_a_Three-axis_accelerometer_for_tilt_sensing,” DFRobot, https://wiki.dfrobot.com/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing

[33] “Taranis X9D+ and X8R receiver binding issue.,” ArduPilot Discourse, https://discuss.ardupilot.org/t/taranis-x9d-and-x8r-receiver-binding-issue/51772/2

[34] “Tutorial: How to flash opentx firmware to radio transmitter,” Oscar Liang, https://oscarliang.com/flash-opentx-firmware-taranis/

[35] “Welcome to OpenTX,” OpenTX, https://www.open-tx.org/

[36] “Taranis X9D plus - FRSKY - lets you set the limits,” FrSky, https://www.frsky-rc.com/taranis-x9d-plus-2/

[37] “How to Flash Taranis internal RF module firmware,” Oscar Liang, https://oscarliang.com/flash-taranis-internal-module/