An Autonomous Search-and-Rescue Helicopter Bound to a Half-Sphere

Anthony VerBurg, Jonathon Chow, Nathaniel Aponte

***Abstract*—We propose and design a simplified autonomous helicopter bound to a half-sphere to explore various control methods and algorithms to pilot the aircraft. This is done in order to develop an autonomous search-and-rescue system. The helicopter has been designed to have two modes of control, manual and autonomous. In manual mode, the helicopter is controlled through the use of a wireless RC remote that allows control of the vehicle from a distance. In automatic mode, the helicopter is controlled with a digital vector PID controller with a bias correction that moves the helicopter to the desired position. The flight hardware to perform this control includes an Adafruit Feather M4 Express for processing input data, performing the controller calculations, and actuating the motor and servo that controls the propellor and pitch respectively; an Adafruit 9-DOF Orientation IMU Fusion Breakout - BNO085 chip for gathering positional information; an RC receiver to perform communication with the wireless RC remote; and a 30A RC Brushless Motor Electric Speed Controller to control the brushless motor. At the moment, the helicopter has achieved level 1 autonomy by Doctor Stuart Kleinfelder’s definitions of autonomy.**

***Index Terms*—Autonomous, Autonomous Flight, Autonomous Search-and-Rescue, Helicopter, S.A.R., Search-and-Rescue**

# I. INTRODUCTION

S

EARCH-AND-RESCUE (S.A.R.) is a time critical operation that requires resources and manpower to carry out efficiently. This is because large areas of land or sea need to be searched to locate a person(s) or object(s). When there is not sufficient manpower to perform S.A.R., autonomous drones and robots could be deployed to help provide additional search capabilities. Choutri, Lahga, and Dala demonstrated a search-and-rescue algorithm using a quadcopter, exemplifying the growing use of aerial technology in S.A.R.[1]. Therefore, it is critical to have reliable and efficient control algorithms to properly navigate the vehicles and robots to achieve their goal in a safe way.[[1]](#footnote-1)

As there are many variables involved in S.A.R., especially in the robotic control and autonomous flight, a small, simplified system can be developed to explore the problem. We propose a small helicopter bound to a half-sphere in order to test and verify control algorithms as well as S.A.R. algorithms. Figure 1 shows the half-sphere coordinate representation that will be used in the development of the control systems of the helicopter. Note that the radius is kept constant, signifying that only two numbers are needed to represent the heliopter’s position in space. Additionally, a small capsule will stand in for humans or objects that the S.A.R. operation wants to retrieve. This simple system will simplify the complexity of 3D control and 3D object locating to a 2D scenario. Once the algorithms are tested and verified there, they can be generalized to higher dimensions, including 3D, and deployed into real scenarios.

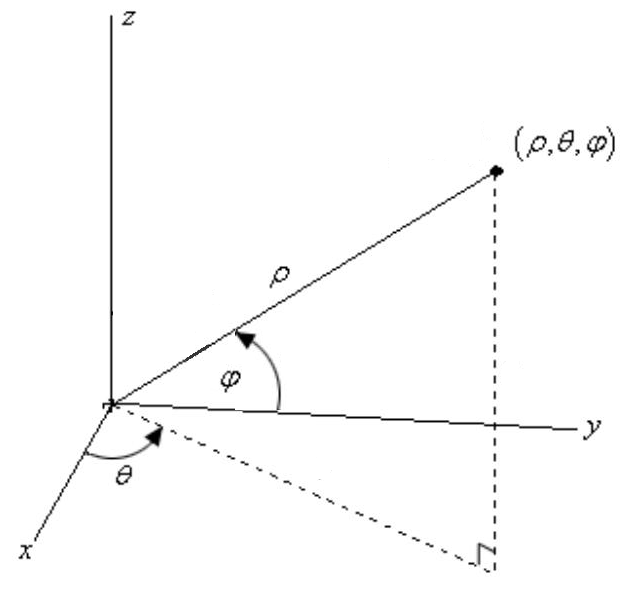


Fig. 1. Spherical coordinate representation of the helicopter’s position in space used by the helicopter’s positioning system

To accomplish the 2D S.A.R. task, algorithms and methods are needed to position and move the helicopter around the half-sphere in a controlled manner. Moreover, S.A.R. algorithms like object recognition and planning are needed to successfully secure the capsule and deliver it. With a large amount of variability in design and technique, a framework was needed to standardize and quantify the performance of the helicopters. Therefore, Doctor Stuart Kleinfelder developed 6 levels of autonomy to quantify the level of control and autonomy that the helicopter system can provide [2]. The levels include:

|  |  |
| --- | --- |
| TABLE 1  KLEINFELDER LEVELS OF AUTONOMY | |
| Level | Description |
| 0 | The helicopter provides manual control capabilities for a human operator |
| 1 | The helicopter can take off, hold its position at a desired height for a few seconds, and then slowly descend in a controlled manner |
| 2 | The helicopter can take off, hold its position at a desired height for a few seconds, perform a 360° rotation around the half sphere at a controlled speed, hold its position at the desired height again, and then slowly descend in a controlled manner |
| 3 | The helicopter can autonomously pick up and secure a capsule in a known location around the half-sphere and deliver it to another known location around the half-sphere |
| 4 | The helicopter can autonomously pick up and secure a capsule from an unknown position around the half-sphere and deliver it to a known location |
| 5 | The helicopter can autonomously pick up and secure a capsule from an unknown position around the half-sphere and deliver it to a known location all while avoiding obstacles in its flight path |

The goal of this project was to achieve all 6 levels of autonomy in order to understand the algorithms and methods required to perform a S.A.R. operation with an autonomous aircraft and build a framework that can scale to larger and more complex systems.

Prior to this helicopter, there have been manually controlled helicopters on a half-sphere, primarily the Vertibird produced by Mattel and the Verti-Pro system produced by Doctor Stuart Kleinfelder [2]. These systems had no autonomy and were controlled by a simple set of levers and an RC remote respectively for throttle and pitch control. There are also quadcopters and drones that are capable of autonomous search-and-rescue operations, but they are not bound to a half-sphere [1]. The proposed system is a hybrid of both of these in order to simplify the system design so that various control algorithms as well as Search-and-Rescue techniques can be explored in a less complex setting.

To develop the helicopter, the system was designed following standard engineering hardware and software methods. For the hardware, we chose to reduce the weight of the helicopter by moving the battery to the base, following a reductionist methodology. Also, we used a servo clamped to the tube which holds the helicopter steady on the half-sphere to tilt the helicopter, simplifying pitch control hardware. For software, we followed a standard finite state machine approach in keeping track of the system state allowing easy manipulation in changing between manual and automatic mode at the press of a button. We also used a standard PID controller on both the “x-axis” and the “y-axis” (as this is a half-sphere, the PID controller is actually controlling the angles of rotation around the half-sphere, but the systems are nearly identical) to keep the helicopter steady when it is at the desired height. Additionally, we used linear interpolation, a standard technique in graphics programming, to move the helicopter gradually. Following this approach allowed for a standard way for controlling the movement of the helicopter.

The results of the design and following the methodology are as follows: the helicopter was 3D printed and assembled together successfully. The electronic subsystems were also wired together and placed within the helicopter assembly. The software has successfully gained digital control over each of the electronic subsystems, being able to poll the orientation sensor for quaternion data, read incoming data from the RC receiver, turn the servo, and throttle the motor. The software can also perform the PID controller calculations. The linear interpolation still needs to be verified that it works and once it is, level 1 autonomy will have been achieved.

# II. Helicopter Design and Materials

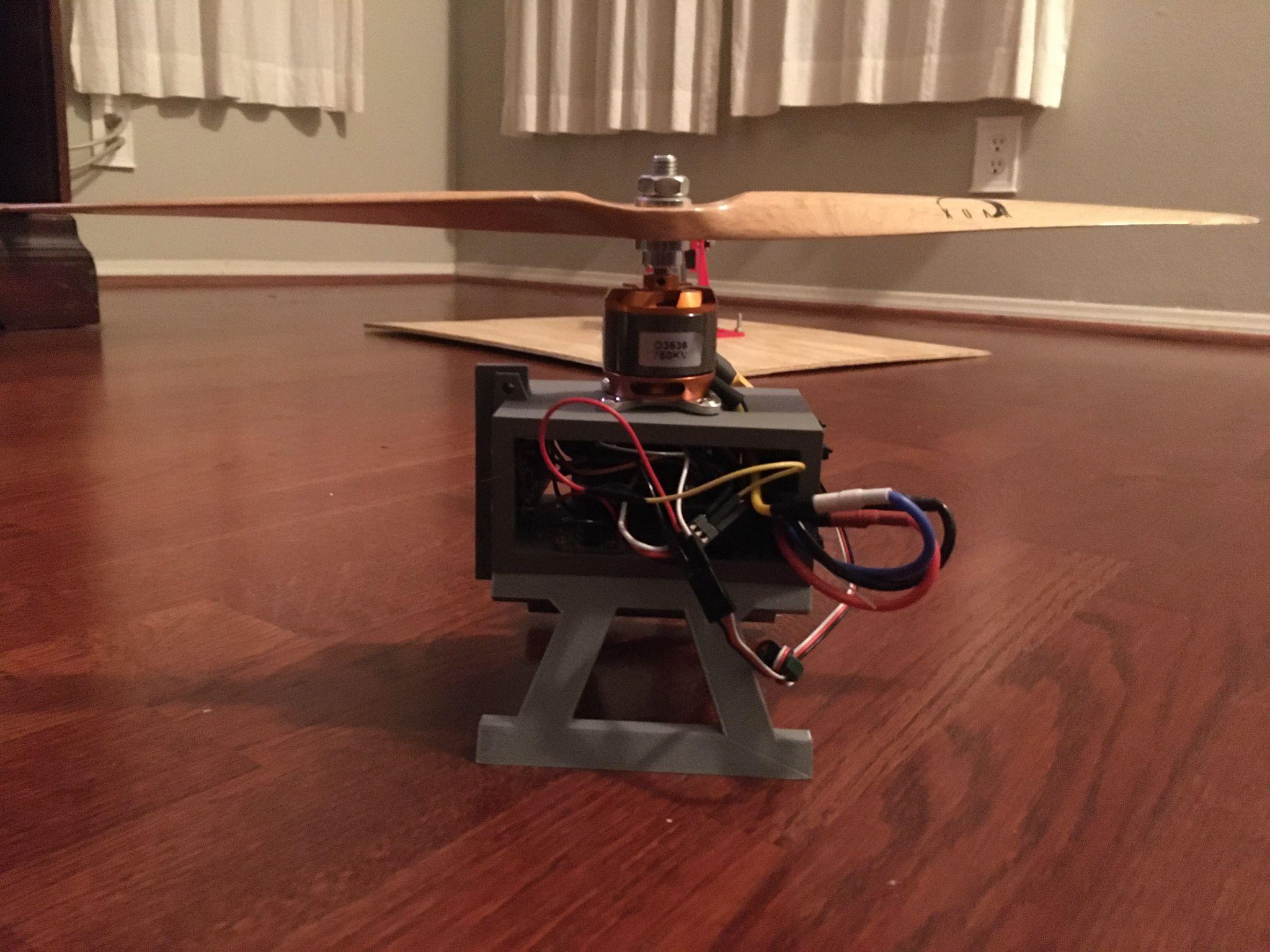


Fig. 2. Design of the Helicopter

For the design of the helicopter in figure 2, we took inspiration from the Verti-Pro system by Doctor Kleinfelder with the use of a base and arm to keep the helicopter within a half sphere range of motion [2]. We further updated his design to include a micro controller and position sensor to achieve autonomous flight. Compared to the Verti-Pro system, we moved all the components into the helicopter, except for the battery. In doing so, we redesigned how to change the pitch of the helicopter with the servo no longer being inside the base. Lastly, we added an enclosed area to house the electronics.

To house the electronics, we went with a box design with a hollow center for the main body of the helicopter. Where threaded inserts are lined up with where we want to place components. The components are attached to the inside of the center casing so parts are not exposed during flight, allowing for minimal damage to components during flight. As for the legs they are two symmetrical pieces attached to the bottom of the body. Splitting the legs added strength and made it easier to 3D print given more orientation options. The orientation used to print allows the legs to land against the grain of the printed material. Larger threaded inserts and screws were used to attach the legs to the center body.

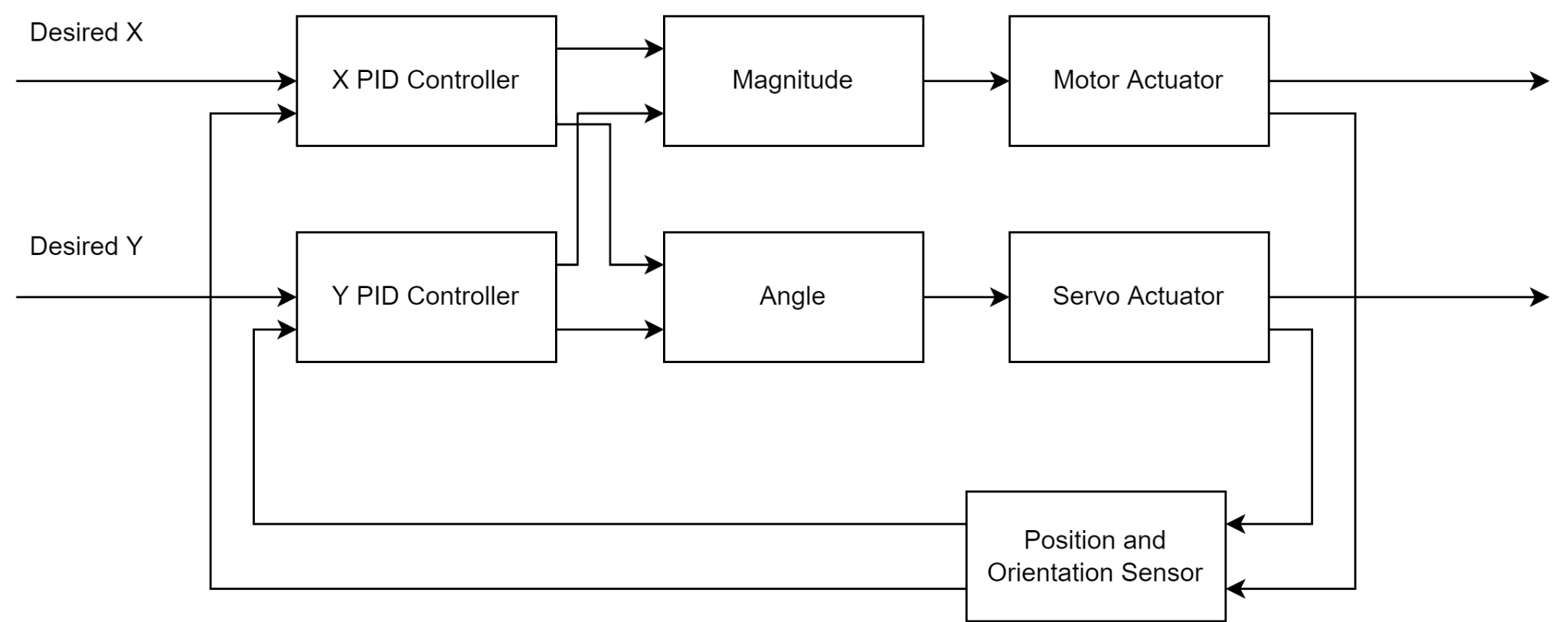


Fig. 3. Vector PID controller to actuate the motor and servo

As for the materials that were used to make the helicopter, we decided to go with polylactic acid (PLA) plastic. Since the majority of the helicopter is being 3-D printed this material offered good durability with a small amount of flex to withstand multiple test flights before breaking. PLA is also biodegradable so it can be recycled after the printed parts are no longer needed. Other materials used are carbon fiber for the torque tube and the propeller having high durability with a small amount of weight. The length of the tube is four feet giving the helicopter a radius of about 4.5 feet keeping sure to leave 10 feet of space when the system is in use.

# III. Helicopter Hardware and Software System

For the main hardware component, we decided to go with the Arduino Feather M4 Express chosen for its small size and performance [3]. Containing 16 PWM output pins in order to control all of our components while also having plenty of flash memory to load up with libraries for hardware components such as the 9-DOF position and orientation sensor. This sensor is used to determine many different variables to be able to calculate values needed for our system such as pitch, yaw, and roll of the helicopter. Which is caused by the Brushless motor and the 20Kg digital servo. The motor is controlled through an Electronic Speed Controller (ESC), while the servo is connected to the Arduino. To power the entire system, we are using a rechargeable LiPo battery connected to the ESC and outputs 5V to power all other components.

In order to control the system remotely another crucial hardware component used is the 4-Channel Remote Control (RC) controller and receiver. We decided to go with a 4-Channel system because this allowed us to implement not only the basic manual controls but also a kill switch and a switch to change between manual and autonomous flight. Which in manual mode directly controls the servo and motor whereas in automatic mode the Arduino microcontroller controls all systems. Although the kill switch can be used in either manual or automatic to stop all systems from continuing to move.

In order to implement the control systems of the helicopter, multiple software subsystems needed to be developed. These subsystems were created with the use of the Arduino IDE and were directly deployed onto the microcontroller. There are currently three main software components that control the helicopter. These are the PID controller pipeline, the system state machine, and the main control loop and device initialization.

The PID controller pipeline currently consists of 3 stages. In the first stage, the position and orientation of the helicopter is calculated using the 9-DOF position and orientation sensor. This information along with the level of autonomy and the current state of the system is used in the second stage of the PID controller pipeline to determine the desired position of the vehicle. This position is determined through a stored route and linear interpolation. Finally, all of this information is fed into the digital vector PID controller to calculate the strength of the motor as well as the angle of the servo. The PID controller is diagrammed in figure 3 and is programmed similar to the approach that Vahid, Givargis, and Miller took for the design of their PID controller [4]. The Y PID controller is slightly modified to work as an integrator to hold the steady state value. Therefore, the Y PID controller is modifying the stored value compared to the X PID controller which is a standard controller. This pipeline is computed when the helicopter is being automatically controlled compared to when it is manually controlled by an operator.

The system state machine is a moore type state machine that manages the mode of the helicopter. The Moore machine is designed similar to how Vahid, Givargis, and Miller designed their digital state machine and took inspiration from Vahid and Lysecky’s Moore Machine [5][6]. Depending on what state the system is currently in, the helicopter will either be automatically controlled by the microcontroller or it will be manually controlled by the wireless RC remote as displayed in figure 4. To transition between states, the system samples the state button on the RC remote using interrupts to detect changes in the button's state. If the button is pressed and the system is in the not pushed state, it transitions to the next state. If it is not and the system is in the pushed state, it also transitions to the next state. Otherwise, the system remains in the current state. After that, the state machine performs the manual or automatic routine depending on the state it is in.

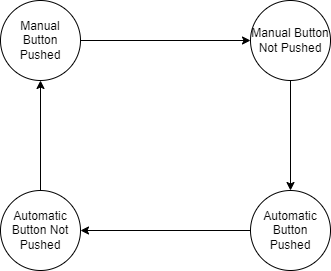


Fig. 4. System state machine of the helicopter mode

The last software component currently implemented is the system initialization and the main control loop. This is a standard Arduino scheme that initializes the system by setting up communication channels between the microcontroller and the various sensors and actuators like the position sensor as well as the electronic speed controller and initializes other important system variables that will be used during runtime. Then the main loop is run. Inside the loop, the results of the state button sampling are used to update the system state machine and then the state machine’s corresponding functions are run depending on the state it is currently in. This loop repeats this action after each iteration completes. As a quick note, the manual control routine reads the incoming signals from the RC receiver and directly dispatches it to the ESC and the servo motor, with no software intervention (other than speed limits) taking place. As the only job of the manual control routine is to multiplex the received signals, it is not considered a major software subsystem.

# IV. Methods

For hardware methods, we chose to put the battery in the base to reduce the weight of the helicopter and the rest of the electronics on the helicopter. Removing the battery and lowering the weight followed a reductionist methodology that optimizes the system for minimal energy consumption during flight. In order to connect the base to the helicopter we use a carbon fiber tube, which we will refer to as the torque tube, that extends out to the helicopter which is attached using a clamp. The clamp is put at the end of the torque tube and mounted onto the servo. The torque tube contains the cable to connect the battery to the rest of the components through the ESC. All components within the helicopter are then connected to the ESC for power supplied through the battery. While also connected to the Arduino which is used to receive inputs and for controlling most components. The Arduino receives inputs from the RC controller and receiver which can control the servo and the motor, while also having a kill switch to stop any moving components and a button to begin the autonomous flying sequence. To control pitch of the helicopter the servo is connected to an out pin on the arduino which rotates the servo which in turn rotates the whole helicopter which causes it to tilt because it is connected to the torque tube which is held in place by the base. To be able to read outputs from the 9-DOF position sensor we chose to go with I2C wiring from the Arduino and the sensor.

For the software methods, we designed a finite state machine for the helicopter to go between manual flight and automated flight. The button on the helicopter controller will change between automated and manual flight when pressed. We also designed a linear PID control system. In the system, we have two PID controllers, one for the x-value and one for the y-value. The values from the two controllers can be used to find the magnitude and angle values. The magnitude value will tell the motor how strong to spin and the angle value will tell the servo how far to rotate. The feedback for the system will come from the position sensor.

# V. Results And Performance

For the Fall quarter, we were able to get all components to receive power from the battery and communicate with the microcontroller. For the position sensor we were able to acquire accurate and real time outputs on pitch, yaw, and roll depending on the orientation of the sensor. Motor connects to the ESC which receives inputs from the microcontroller to be able to spin the motor. The servo also receives input from the microcontroller to be able to turn given degrees depending on the input. Both the servo and the motor can receive input from the RC remote which sends its signals to the receiver which goes into the microcontroller then outputted to the motor and servo. We use software interrupts in order to improve performance and responsiveness of the manual controls with the RC remote. All parts fit on the 3-D printed parts that we designed through CAD.

For the Spring quarter, we updated the design so that the helicopter will be more stable and more structurally sturdy. The previous fall quarter design cracked easily and suffered from the vibrations caused by the motor. The new design is more rigid and will not crack under the vibrations. Additionally, we had the helicopter’s first flight. We then tuned the helicopter’s PID controller to make it a lot more stable during flight. We have nearly verified that the interpolation works and will achieve level 1 autonomy for the helicopter. Sadly, we had a lot of setbacks this quarter, with having to reprint the helicopter on multiple occasions, thus making it impossible to go into further levels of autonomy due to time constraints. The code will be made public on github though, so the other levels of autonomy may be achieved in the future.

VI. Conclusion

For this project, an autonomous helicopter bound to a half-sphere was designed. The helicopter was designed in Solidworks and 3D printed in PLA. The choice of lightweight electronic hardware like the Adafruit Feather M4 Express allowed for reduced weight to optimize the system for flight as well as a compact 9 DOF position and orientation sensor. Additionally, the choice of a LiPo battery provides the system with an adequate supply of energy to run the system. The software of the system was designed to support multiple modes of operation, which are manual and automatic mode at the current moment. The system is controlled with a digital vector PID controller that digitally integrates the y PID output to create a hysteresis effect. Altogether, the system has full control over each of the electronic hardware subsystems and can gather data from the sensor and can command the actuators throttle and pitch.

In the fall quarter, the project mainly consisted of CADding the helicopter and putting it together. In the spring quarter, the helicopter was redesigned to be more stable and not crack from the vibrations. The helicopter was also programmed for manual and automatic flight. The helicopter can manually fly and land and level 1 autonomy was achieved.

To extend this project, one can test other different information gathering methods, like LIDAR, to perform object recognition with. Also, the project can be extended by providing it with networking capabilities so that it can communicate its flight path with a computer, so that it can be displayed for human control operators to monitor the system with. Another enhancement is that multiple S.A.R. algorithms can be tested with model people instead of a capsule to verify their effectiveness for locating people in distress.

Appendix 1

Many standards were chosen to aid in the rapid development of the helicopter. The standards chosen are the use of the Arduino IDE for software development, the use of standard pulse width modulation communication with the RC receiver and ESC, the use of I2C for communication with the 9 DOF position and orientation sensor, and the use of standard M3 screws for assembling the helicopter.

The choice of using the Adafruit Feather M4 Express allowed us to use the standard Arduino IDE to develop the software with. This allowed us, who were already comfortable with Arduino, to get started on the project relatively quickly. Also, the choice of using a standard RC receiver with pulse width modulation communication allowed for the code to work even when the receiver ended up being swapped out for a different one when we got a different RC remote. The same applied for communication with the ESC using pulse width modulation. The choice of using I2C for communication with the position and orientation sensor allowed us to use a prebuilt library to query the chip for the orientation. Finally, the choice of using M3 screws (except for the electronics that needed M2 screws) for assembling the hardware guaranteed that when the holes of two parts of the hardware were lined up, they were guaranteed to fit properly and connect firmly with the screw.

The design of the system complied with following all of these standards. The Arduino IDE was used for all software development of the system, pulse width modulation with standard widths were used to meet all specifications for the electronic subsystems, I2C was used and wired correctly to communicate with the position and orientation sensor, and M3 screws and screw holes were used in the assembly of the helicopter.

Appendix 2

One of the challenges we had was controlling the motor. When we were first testing the motor, we accidentally ran the motor at full speed and burnt it out. We had to buy a new motor and put a limit on the speed. We started at a really low speed limit in the beginning when retesting the motor. We slowly increased the speed of the motor while testing to find the right speed. Once we find we find the right speeds, we can have the helicopter hover and change heights without going out of control.

Another challenge we had was the screw holes not lining up when we were putting the helicopter together. When measuring the distance between the holes, we measured the distance between the closer edges of the holes. When we were cadding the helicopter we used this distance at the distance between the further edges so the holes were too close. We had remeasured the distance and drill new holes for them to line up.

While programming the control system for the helicopter, the X controller would dominate the power of the motor when the error got large. We compensated this by weighting the contribution of each term and decreasing the weight of the X component. We also integrated the Y component to allow it to retain a steady state when there was no error.

Appendix 3

When cyber security issues are encountered, they are typically associated with the system having WiFi or bluetooth capabilities. However, the system does not have any WiFi or bluetooth capabilities, so it does not have any potential security problems in regards to these domains. Although, one potential security issue we may face is with the RC remote and receiver. Since the remote and receiver communicate through a 2.4 GHz band, another remote transmitting using the same frequency as our remote can possibly gain control of our system. This attack is known as spoofing. Spoofing can be combated by requiring a means of authentication so that access is restricted to only certain parties. In the case of our system, we could require the remote to send a digital signature before sending any data to enforce the authentication restriction. Furthermore, so that other parties do not intercept this digital signature, the signature can be encrypted using asymmetric keys and sent to the receiver so that only the receiver can decrypt it and verify the remote. In this way, the system would prevent any spoofing from other external remotes. As the RC receiver is the only software entry point, there would be no other security issues in this area.

A potential hardware security issue is that the port to the microcontroller is not protected in a secure way. This means that if a potential attacker could get to the helicopter, they could upload any program they wish into this system by simply connecting their computer to the microcontroller with a cable. A potential solution is to enclose the microcontroller and other electronics in a locked container on the helicopter and only those with a key could access it and upload control software. In this way, the locked container would limit access to only parties who are allowed to modify the system software.

Acknowledgment

We would like to thank everyone who participated in the creation of the system, including Nathaniel Aponte for his contributions to the hardware design of the helicopter and research to getting the system functioning, Jonathon Chow for his contributions to the hardware design as well as his contributions to the design of the PID controller, and Anthony VerBurg for his contributions to the design of the software systems including the state machine and the PID controller. Without their help, this project would not have been accomplished. We would also like to thank our advisor, Doctor Stuart Kleinfelder, for providing materials, the electronic equipment, and insights into the development of the system as well as arranging the autonomous flight competition. Additionally, Doctor Kleinfelder provided various sources of funding so that the helicopter could be manufactured and assembled. Without his contributions, the project would not have been possible. We would also like to thank Fabworks for providing access to their 3D printer and performing the 3D print of the helicopter. We would also like to thank the University of California, Irvine Department of Electrical Engineering and Computer Science for providing a space to work on this project as well as providing Doctor Kleinfelder resources for the EECS 159A Senior Design course, which we have greatly benefited from.

References

[1] K. Choutri, M. Lagha, and L. Dala, “A fully autonomous search

and rescue system using Quadrotor UAV,” *International Journal of*

*Computing and Digital Systems*, vol. 10, no. 1, pp. 403–414, 2021

[2] S. Kleinfelder, “The 2021 EECS 159 Autonomous Flight

Competition.” .

[3] K. Rembor, “Adafruit Feather M4 CAN Express,” *Adafruit*

*Learning System*, 25-Jan-2021. [Online]. Available:

https://learn.adafruit.com/adafruit-feather-m4-can-express. [Accessed: 02-Dec-2021].

[4] F. Vahid, T. Givargis, and B. Miller, “Control Systems,” in

*COMPSCI 145/CSE 145: Embedded Software*, Zyante Inc., 2015.

[5] F. Vahid, T. Givargis, and B. Miller, “State Machines,” in

*COMPSCI 145/CSE 145: Embedded Software*, Zyante Inc., 2015.

[6] F. Vahid, “Sequential Logic,” in *COMPSCI 151: Digital Logic*

*Design*, R. Lysecky, Ed. Zyante Inc., 2015.

1. This work was submitted on March 10, 2022. This work was supported by the University of California, Irvine Department of Electrical Engineering and Computer Science

   Anthony VerBurg is with the University of California, Irvine, Department of Electrical Engineering and Computer Science, 2200 Engineering Hall Irvine, CA 92697-2625 USA (e-mail: verburga@uci.edu).

   Jonathan Chow is with the University of California, Irvine, Department of Electrical Engineering and Computer Science, 2200 Engineering Hall Irvine, CA 92697-2625 USA (e-mail: chowjh3@uci.edu).

   Nathaniel Aponte is with the University of California, Irvine, Department of Electrical Engineering and Computer Science, 2200 Engineering Hall Irvine, CA 92697-2625 USA (e-mail: aponten@uci.edu). [↑](#footnote-ref-1)