December 16, 2018

Project: Autonomous Indoor Drone

Team: Henry O'Meara and Jeffrey Wen

Final Project Report

Dear Dr. Barrett:

The following paper contains the final project report detailing the background, design, progress, cost, and timeline of our project. The project design is detailed with specifications and goals as well as a schematic and functional block diagram. The progress of this project is demonstrated through current and alternative solutions that have been considered to achieve the project goals. A preliminary cost analysis details how much has been spent on the project and the source of funding. Finally, a project timeline presents self-imposed deadlines that we aim to follow to ensure successful completion of this project. The report has been viewed by both the project members as well as the project faculty advisor.

Best Regards,

Jeffrey Wen

Henry O'Meara

Autonomous Indoor Drone

EE4820 Senior Design I

Final Project Report

Jeffrey Wen and Henry O'Meara

Electrical and Computer Engineering Department

College of Engineering

University of Wyoming

Laramie, WY 82071

jwen@uwyo.edu

homeara@uwyo.edu

Abstract

The need for autonomous indoor drones arises for search and rescue, inspection, and mapping purposes. This project aims to address these needs by developing a drone capable of autonomously navigating an indoor environment using a combination of sensors. The Intel Aero Drone was selected for this project because of its appropriate size for indoor flight, ability to run Ubuntu, and included stereo camera. In addition to the stereo camera, a 2D horizontal distance sensor and downward-facing distance sensor were added to the drone to increase the drone's spatial awareness in non-forward directions. This combination of sensors provides the necessary hardware to attain autonomous flight. To control the drone, a Python program is being developed that utilizes MAVLink messages and DroneKit functions to send commands to the flight controller. So far, testing has shown that the drone can be controlled autonomously with this method. The next step is to integrate the stereo camera and distance sensors into one autonomous control scheme.

Keywords: drone, UAV, autonomous, Intel Aero, indoors

Overview

The goal of this project is to develop an unmanned aerial vehicle (UAV) capable of autonomously navigating an indoor environment. The need for this technology stems from possible applications of indoor UAVs which include search and rescue, inspection, and mapping. While UAVs could possibly be manually controlled, the need for autonomy arises in order to navigate more efficiently and safely than a human controller. However, most autonomous UAVs have employed the use of the global positioning system (GPS) to aid with navigation, but GPS signals are too weak and unreliable to be used for indoor flight. To compensate for the lack of GPS, this indoor autonomous UAV will use additional sensors to localize itself and map its surrounding environment. A stereo camera capable of calculating depth information will be located on the front of the UAV, and additional depth sensing capabilities will be added to detect obstacles in non-forward directions. Interfacing these sensors with the flight controller and onboard central processing unit (CPU) will allow the UAV to make informed decisions to attain autonomous flight. Several technical specifications are outlined to quantify and track the success of the project. As a whole, the UAV will be considered a success if it is able navigate and map the fifth floor hallway of the University of Wyoming engineering building and return to its starting point in a single battery life. The Intel Aero Ready-to-Fly Drone was chosen for this project, and work has begun to get the drone to takeoff off indoors. Though the drone is now able to be programmed to takeoff indoors, there are still many steps necessary before the drone can safely navigate an area.

Since a ready-to-fly drone was purchased, the scope of this project will be focused on programming the drone to execute flight commands and integrating sensors to better aid the

autonomous navigation. The project, thus, is much less focused on the mechanical build of the drone but rather the software development of an existing drone system to accomplish autonomous capabilities. It will require a proper understanding of sensor interfacing, and a methodology for integrating the sensor information into an autonomous program.

This report will cover the details, the current progress, and the future plans of the project. First, the background research into the fundamentals of drone design will be discussed in order to set a basis for proceeding material. This will cover the communication system of current drone technology as well as an overview of the Intel Aero drone and why it was chosen. Then, a technical project description will be provided to detail the inner workings of the project, the steps taken to achieve the current progress, and the future work that needs to be accomplished for success. The report also covers other considerations such as the cost and ethical concerns of the project. The appendix includes a schedule of deadlines to complete certain tasks, a full list of parts, a schematic for the downward-facing sonar, and a construction diagram of the various parts.

Background

The structure and control of drones have become quite standardized for the convenience of hobbyists and researchers alike. All UAVs contain a flight controller (FC) which is responsible for taking commands from a remote controller or onboard computer and executing the command. The flight controller controls the speed of the propellers, and it takes sensor readings from the inertial measurement unit (IMU) and barometer to adjust its flight to fit what the user has specified. The flight controller contains an autopilot firmware that handles the

control algorithms of the aircraft. The two main autopilot firmwares are Ardupilot and PX4. The flight controller communicates with other devices such as a ground control station (GCS) or onboard computer using a protocol called MAVLink. All remote aircrafts are considered micro-aerial vehicles (MAVs), and MAVLink has become the primary communication protocol for these devices [1]. To make the protocol more user friendly, there have been several higher-level abstractions created for developers. Dronekit, an application program interface (API), is a python-based method for programming onboard computers to send valid instructions to the flight controller [2]. MAVROS, a robotic operating system (ROS) package, was developed for the purpose of making the programming of a drone more compartmentalized and higher-level as well. These fundamental concepts of drone development will be the building blocks of this project.

The Intel Aero Ready-to-Fly drone is a UAV designed specifically for developers who want to begin implementing their drone project without having to build a custom drone from scratch [3]. The Intel Aero was ideal for this project's specific needs and was chosen for several reasons. In looking for a drone, we needed three specifications: the ability to have flight commands programmed using an API or Standard Development Kit (SDK), a small enough size to be able to fly through doors, and a customizable frame meant to carry additional payloads. Specifically, a drone with a maximum diameter of 0.75 meters and extra payload capacity of at least 250 grams was required. With these considerations, the drone selected for this project was the Intel Aero Ready-to-Fly Drone. This UAV was selected because it met all of our required specifications for a development drone and included a stereo camera, the Intel RealSense R200; a flight controller, an STM32 microcontroller; and an onboard computer, the Intel Aero Compute

Board. The inclusion of all of these features with a pre-built drone was ideal for spending less time on finding additional, compatible components like the stereo camera and onboard computer and instead, being able to jump right into the process of developing autonomous indoor flying capabilities. The Intel Aero Compute Board is able to run a full version of Linux Ubuntu 16.04 and supports drone development abstractions including Dronekit and MAVROS. It is also a medium-sized UAV with a diameter of 0.59 meters, perfect for navigating indoor spaces, and it has enough power and a customizable frame that allows for adding additional sensors. Other drones were considered including the DJI Matrice 100 and the line of Parrot drones. The Parrot drones were not suited for adding sensors, and while the Matrice 100 fit all of our desired specifications, it was $2000 more expensive than the Intel Aero and did not include a stereo camera. After a thorough search of commercially-available drones, the Intel Aero came through as the most cost-effective platform for developing our project.

Technical Project Description

The final product of this project will be a UAV capable of navigating through an indoor space autonomously. To achieve this, there are several stages that the UAV will operate in during its flight. First, a user should check the safety of the environment before sending the start command to the UAV to enter autonomous mode. After receiving the start command, the UAV should arm its motors, takeoff, and hover at an appropriate height. The UAV should then begin to map its surroundings and determine the direction of greatest uncertainty or openness. Next, the UAV should proceed in the direction of openness and map its surroundings. Upon reaching the boundary of the local environment, the UAV should return to its takeoff position while possibly

exploring any unmapped areas on its way back. At any point in this sequence of operations, should the user determine that autonomous flight is no longer safe, a manual override could be sent to the UAV to land. While in operation, the UAV should determine the location of obstacles and avoid any potential dangers. Ultimately, the final product will be a UAV that can safely and autonomously navigate an indoor space.

Given that the chosen drone for this project is the Intel Aero, there are several quantitative specifications that should be met to consider this project a success. The first requirement is that the drone will have a minimum flight time of 12 minutes. The projected hover time provided by the datasheet is 20 minutes, which does not account for the active movements of a UAV nor any extra payload. Thus, setting a base of 12 minutes is a reasonably achievable design requirement that also gives an adequate time for the UAV to complete a mapping. Additionally, the UAV and all of its components should be at or below the weight of 1800g. The maximum takeoff weight is 1900g according to the data sheet. Currently, the drone and the battery together weigh 1302g together, leaving 498g of leeway for additional sensors. The 1800g payload, therefore, gives an attainable final weight that leaves a safety margin of 100g to ensure proper flight mechanics and flight efficiency for flight time considerations. In regards to safety, the UAV should operate within a safety margin (distance from any object) proportional to the width of the local environment with an absolute minimum of 0.3 meters from the edge of the propeller guards. This distance provides a safe area for the UAV to operate without limiting the size of space that it can safely fit through to exclusively open spaces. The light detection and ranging (LiDAR) system that will be used, the RPLidar A2, has an operating distance of 0.15 meters to 18 meters. With the LiDAR placed at the center of the drone,

approximately 0.3 meters from the edge of the propeller guards, it should have no problem detecting distances just beyond the propeller guards. Lastly, as a holistic technical requirement, the UAV should be able to navigate and map the fifth floor of the engineering department and return to its starting position within one battery life.

The drone has several pieces of hardware that must be integrated together to make the drone capable of autonomous flight. The main code for operating the drone is located on the onboard computer. The onboard computer is connected to the other sensors via the field-programmable gate array (FPGA) which allows the sensor pins to change functionality if needed. The compass and downward-facing sonar are connected in this manner and provide the onboard computer with information about the direction and height of the drone. The schematic for the downward-facing distance sensor can be found in Appendix 3. This information is also routed to the flight controller, so the autopilot firmware is able to use that data for flight stability and control algorithms. The LiDAR is separately connected to the flight controller using the universal serial bus (USB) 3.0 port, and the stereo camera has its own dedicated USB 3.0 connection to the onboard computer. The program on the onboard computer will use data from the stereo camera, LiDAR, and sonar to determine the flight plan required for safe autonomous navigation. The onboard computer will then forward the flight commands to the flight controller. The flight controller is directly connected to the IMU, and using the data from the IMU, compass, and sonar, it determines the speed of the motors required to achieve the received commands. Finally, the flight controller sends the proper signals to the motors to operate at the desired speeds. The functional block diagram of the Intel Aero Drone and its connected sensors can be found in Appendix 4.

Several steps have already been taken to work towards completing the project. The first thing that was accomplished was setting up the drone for development. Out of the box, the drone required images to be updated and flashed for the operating system (OS), basic input/output system (BIOS), field-programmable gate array, and flight controller. For the autopilot software on the FC, Ardupilot was chosen instead of the preinstalled PX4 because of the availability of documentation and the general consensus online that Ardupilot is slightly more stable than its counterpart. Then, a laptop was connected to the Aero's wifi access point. Using a ground control station called QGroundControl, the drone's accelerometer and compass were calibrated, making the drone flight-ready. Next, the drone was taken to an outdoor open area with Vic Bershinsky to do a manual test flight and make sure all the components were working correctly. The drone flew well for approximately five minutes before encountering an Extended Kalman Filter (EKF) error, meaning there was a sudden change that the control system could not correct for. After analyzing the situation, it was concluded that the battery must have slightly slid out of its compartment and caused a sudden shift in weight that brought the drone down. The battery was only held in by two Velcro straps. Even when fully tightened, the battery was still able to slide out of the back of the drone fairly easily. To prevent this, a piece of foam was inserted between the battery and straps to provide more friction. The second test flight went without any issues after the foam addition. The drone came preinstalled with a terminal-based Linux operating system called Yocto. While this OS is usable, it is much less developer-friendly as there is no visual desktop. To make prototyping and development easier, Ubuntu 16.04 was installed on the onboard computer. After the Ubuntu installation, the BIOS, FPGA, and FC images had to be flashed again.

With the drone setup for development, we began exploring the various methods of controlling the drone with self-written programs. We decided to first look at the Python-based API, Dronekit, because it seemed to offer a means of quickly prototyping a code to get the drone to takeoff. ROS was also looked into but required too high of a learning curve that would need extensive amounts of time. ROS did appear to offer a much more organized method for combining all of the components like the stereo camera, onboard computer, FC, and ground control station, so we hope to integrate our Dronekit design into ROS once we have more time to learn the system. While Dronekit did offer a higher-level abstraction of MAVLink, a general understanding of MAVLink was still necessary and recommended to comprehensively develop in Dronekit. Thus, significant time was spent learning the basics of MAVLink messaging and Dronekit functions. Though there was extensive documentation and examples on using Dronekit, nearly all of it was oriented on flight modes that depend on a GPS signal, specifically Guided mode. Guided No GPS mode was recently created for developers in GPS-denied environments, but limited documentation was available. After trying several different methods, we were finally able to control the motors using the set_attitude_target messages, which allowed us to specify the pitch, roll, and yaw rate as well as the thrust. We wrote a simple script to have the drone takeoff until it reached one meter, hover for three seconds, and then land. The drone was tied down to anchors in order to prevent too much drift. In the test, the drone was constantly attempting to go higher because the barometer gave unpredictable values when the propellers were spinning. The drone was successfully able to land, however.

The initial indoor test showed that the barometer was not consistent enough to use indoors because of erroneous altitude readings during takeoff. Barometer readings were accurate

when propellers were not installed on the drone, but when the propellers were installed, the added airflow affected the barometer readings in an indoor environment. Therefore, alternative altitude measurement methods must be considered. Three possible alternatives are being evaluated. The first option is to rely solely upon computer vision methods using the preinstalled downward-facing camera. If objects of a known size were present on the floor (e.g. tile floor), the drone could takeoff until a shape only occupied a certain number of pixels in the camera's field-of-view. However, this method would be the least practical, as it would not work in an unknown environment. The second method is to deflect some LiDAR scans from the horizontal 2D LiDAR sensor downward to measure altitude. This LiDAR sensor is mounted underneath the body of the drone in the middle of its four legs. Since some angles of measurement will be blocked by the legs of the Aero anyway, a mirror mounted on one of the legs could deflect these lasers for altitude measurement. It is not yet clear if these measurements could be routed to the FC as the primary altitude measurements used to stabilize the drone and hover. The third method is to attach a downward-facing, single-direction sonar sensor that can be set to the main altitude measurement method in QGroundControl. This method is the most likely to be successful but would require additional hardware. One of these methods will be utilized to attain a precise hover altitude after takeoff. Once this goal is achieved, the next steps to attain autonomous flight will be using data from the distance sensors and stereo camera to send commands to the flight controller to control the pitch, yaw, roll, and thrust of the drone.

As previously mentioned, a 2D LiDAR scanner will be used to assist with localization of the drone and possibly mapping. At first, the Aero would not recognize the chosen LiDAR device, RPLidar A2, when it was connected to the onboard computer's USB 3.0 port. It was

determined that the Ubuntu kernel on the Aero did not include the necessary driver to interface a universal asynchronous receiver/transmitter (UART) device with the USB 3.0 port. Therefore, the CP210x USB to UART Bridge Virtual COM Port driver was installed using the guidelines presented in [4] which then allowed the LiDAR to successfully communicate with the drone. Testing this LiDAR scanner with the Aero yielded promising results. The LiDAR was able to measure distances of at least 12 meters, and the BreezySLAM Python library was utilized to generate a 2D map of a room and localize the scanner in the room as it moved around. Further testing is necessary to determine if this method of simultaneous localization and mapping (SLAM) is accurate enough over longer distances such as a hallway.

Other Considerations

This project's expenses are for a new drone, the Intel Aero, and the necessary accessories to power the drone and increase its sensing capabilities for autonomous flight. If another group were to try to recreate this project in the future with a new drone, not all of these expenses would be necessary. For example, the battery charger and power supplies for the Aero could be used again. However, the majority of the cost for this project comes from the Intel Aero Drone itself and the 2D LiDAR scanner. Each component of this project and its cost can be found in the parts list in Appendix 2. The total cost of this project will be about $1610.20. This project is funded by a research grant from Dr. Suresh, and funding has already been secured.

There are many aspects that must be considered to deem this project ethical. Social impacts of indoor autonomous drones may include military applications. Such drones may eventually be capable of inspecting a building for people or explosives prior to sending soldiers

into harm's way. This could save lives of soldiers who would normally have to enter a building without any prior surveillance. However, with all military technology, there is potential for misuse or the technology falling into the wrong hands. If this technology would be more likely to save lives, then the next thing to consider is the economic impact of indoor autonomous drones. There are no apparent negative economic impacts of this technology. However, these drones could prove to be a more economical way to inspect or map buildings that may not be structurally stable after a fire, for example. The project should hold no potential for affecting the political landscape since it presents no political applications. Additionally, these same applications demonstrate the innate health and safety aspects of indoor autonomous drones. This project is intended to allow drones to fly in areas not safe for humans, so as long as testing is done in a closed environment without people, then this project will be safe and healthy. The main environmental concern associated with drones is noise pollution. These drones are intended to fly where it is not safe or easily accessible for humans, so the generated noise would be a non-issue. The main societal impact of this technology is associated with its application for aiding with search and rescue. This mission is inherently good, and once again this specific engineering solution should be deemed ethical and responsible. Since this drone is flown away from people and has a utilitarian mission, aesthetics of the drone do not need to be considered as long as the additional parts are securely attached. If this product ever needed to be manufactured on a large scale, it is important to design for manufacturability. Since the Intel Aero Ready-to-Fly Drone comes assembled and will only need two additional hardware additions, the 2D LiDAR and downward-facing distance sensor, this product can easily be reproduced. Additionally, the drone is designed to be reusable with rechargeable batteries which means this project is also

sustainable. Finally, since the Federal Aviation Administration (FAA) does not regulate indoor unmanned aircraft systems (UAS), there are no governing standards that need to be followed.

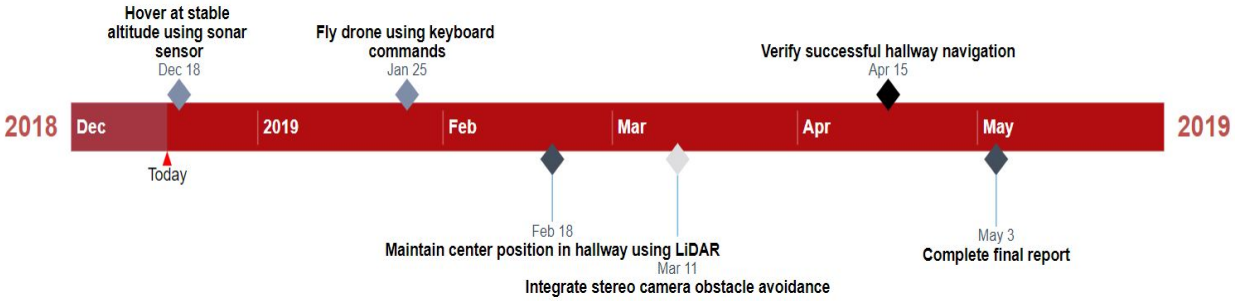
References

1. “MAVLink Developer Guide.” *MAVLink*, https://mavlink.io/en/guide/. Accessed 5 Nov. 2018.
2. “About DroneKit.” *Dronekit*, http://python.dronekit.io/. Accessed 25 Oct. 2018.
3. “Intel Aero Wiki.” *Github,* https://github.com/intel-aero/meta-intel-aero/wiki. Accessed 1 Oct. 2018.
4. “RPLidar A2 Setup on Linux.” *Github*, 6 Jan. 2018, http://samliu.github.io/2018/01/06/rplidar-a2.html. Accessed 8 Nov. 2018.

Appendices

1. **Schedule**

Task	**Completion Date**
Hover at stable altitude using sonar sensor	December 18, 2018
Fly drone using keyboard commands	January 25, 2019
Maintain center position in hallway using LiDAR	February 18, 2019
Integrate stereo camera obstacle avoidance	March 11, 2019
Verify successful hallway navigation	April 15, 2019
Complete final report	May 3, 2019

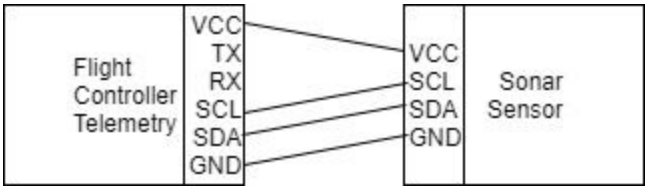


2. **Parts Lists**

- Intel Aero Drone ($1099.00)
    - Purchased from: Amazon.com
- USB 3.0 OTG cable ($16.00)
    - Purchased from: Amazon.com
- Anker 4-Port USB 3.0 Hub ($9.99)
    - Purchased from: Amazon.com
- 5V Power Supply for Intel Compute Board ($6.95)

    - Purchased from: Amazon.com
- 12V Power Supply for Drone ($9.29)
    - Purchased from: Amazon.com
- Power Supply for Drone Adaptor ($7.99)
    - Purchased from: Amazon.com
- 4000mAh 4S LiPo Battery ×2 ($84.58)
    - Purchased from: Amazon.com
- Smart LiPo Battery Charger ($49.95)
    - Purchased from: Amazon.com
- RPLidar A2 ($299.00)
    - Borrowed from: Dr. Suresh
- 3D-Printed Propeller Guards ($12.00)
    - Purchased from: Trevor Trouchon
- Screws and Washers ($15.45)
    - Purchased from: Ace Hardware

3. Circuit Diagram of Downward-Facing Distance Sensor



4. Construction Diagram