Long Exposure Light Painting Utilizing a CrazyFlie Drone

Noor Ansari, Audrey Godsell, Seongyong Hong, and Shirley Shah
Department of Aerospace Engineering, University of Illinois at Champaign-Urbana

The goal of this project is to outfit a CrazyFlie drone with the Bitcraze LED-ring deck and fly the drone in a pattern such that it spells out different input messages. These messages will be read when a long exposure photo is taken of the drone flight. This goal is motivated by the University of Illinois's Engineering Open House (EOH). Every year, the engineering department hosts EOH, an event where members of the general public can explore the interesting projects the college is working on. Clubs and research groups demonstrate their various projects as well as engage with the public and teach them engineering fundamentals. The team will apply for a "painting" with drones exhibit for Women in Aerospace (WIA) at this year's EOH using our controller and drones. Major tasks in this project include setting up a controller and observer that can be used in a dark room, creating paths for each letter of the English alphabet, creating a script to take in letters and output the path the drone will fly, and configuring the LED-ring deck to turn off between spelling out letters. Because the drone will be flown in the dark, the team will use the Bitcraze lighthouse positioning system to determine our drone's position. Throughout the course of this project the team was able to recreate the dynamics model and controller for the drone that accounts for the added mass from the LED ring and lighthouse deck. The team was also successful in incorporating the lighthouse positioning system into the observer. Ultimately the drone was successful in legibly writing "NOOR" in long-exposure photography. However the controller used on the drone could be further tuned to allow the drone to move with greater precision.

I. Nomenclature

AE	=	Aerospace Engineering
EOH	=	Engineering Open House
EOM	=	Equations of Motion
IMU	=	Inertial Measurement Unit
LED	=	Light Emitting Diode
RMSE	=	Root-Mean-Square-Error
a_z	=	IMU's acceleration measurement in the x-axis
g	=	Gravitational acceleration (in m/s^2)
J_x	=	Moment of Inertia about x
J_y	=	Moment of Inertia about y
J_z	=	Moment of Inertia about z
k_F	=	Force Coefficient k_M
Moment Coefficient i	=	Input vector
LQR	=	Linear Quadratic Regulator
lh_x	=	x position data from lighthouse
lh_y	=	y position data from lighthouse
lh_z	=	z position data from lighthouse
m	=	mass
o	=	Output vector
O	=	Position vector
o_x	=	x position of CrazyFlie
o_y	=	y position of CrazyFlie
o_z	=	z position of CrazyFlie
p	=	Parameter vector

s	=	State vector
$\hat{s}$	=	State estimation vector
u	=	State vector of linearized system
x	=	State vector of linearized system
$\hat{x}$	=	State estimate vector of linearized system
y	=	Output vector of linearized system
ψ	=	Yaw (in radians)
θ	=	Pitch (in radians)
ϕ	=	Roll (in radians)
v_x	=	Velocity in x direction (m/s)
v_y	=	Velocity in y direction (m/s)
v_z	=	Velocity in z direction (m/s)
w_x	=	Angular velocity about x-axis (rad/s)
w_y	=	Angular velocity about y-axis (rad/s)
w_z	=	Angular velocity about z-axis (rad/s)
ω	=	Angular velocity vector

II. Introduction

Drones are increasing in accessibility and prevalence in today's society and are being used for increasingly more artistic pursuits. Drone light shows utilizing hundreds of drones can create holographic-like images in the sky. A single drone can create art as well. This paper details the development of a CrazyFlie drone modified to aid in creating long-exposure pictures that spell words. The team will use a CrazyFlie, lighthouse deck, and LED light ring to accomplish this task. This task is motivated by wanting to do an artistic light painting demonstration for Engineering Open House (EOH) showcasing the use of control theory in Aerospace Engineering.

This final report includes four distinct sections. First, the theory behind the development of the custom controller and observer for the team's drone will be discussed. After the theory section, report discusses the additional hardware that was require to conduct this project. The project utilizes an LED ring light[1] mounted on the CrazyFlie Drone. While the drone itself has LED lights built into its structure, the ring lights provide a brighter source of light, improving the contrast of the light-painted letters. The underlying principles and components of the lighthouse system's operation is also discussed.

Next, the report discusses implementation of the additional hardware, lighthouse system, and the flight paths. The changes in the drone's characteristics due to the additional hardware attachment is discussed. The changes to the drone's firmware code to read and log the lighthouse system's measurements is discussed. The report also describes the process utilized to create letters for the drone to draw. Each letter of the English language was coded into a path for the drone to follow. By concatenating the paths, the drone can be instructed to spell out short words.

Finally, the report concludes with a discussion of the results. The final long-exposure pictures will also be presented in the results section.

Previous AE 483 teams have accomplished similar projects. Last year, a group drew out the Illinois block I shape using a CrazyFlie using a long-exposure photo of their drone flight[2]. Our project expands on this by writing multiple letters in a row, while turning the mounted lights on and off as needed. Instead of drawing a single shape, the team aim to spell words. Another past report uses a motion capture system to draw out shapes horizontally. This project discussed the difficulty of implementing the motion capture system[3]. Our team's usage of the lighthouse deck allows for portable precision tracking, which differentiates us from this previous project. The makers of CrazyFlie provide some guidance on light painting with the lighthouse deck[4]. Our project expands on this by spelling words instead of randomized shapes.

III. Theory

A. Equations of Motion and Measurement Equations

The behavior of a dynamical system such as the CrazyFlie drone can be characterized by a set of equations of motion (EOM), which describes the relationship between the time rate of change in the system's state $\dot{s}$ and current state s, control inputs i, and relevant parameters p.

$$\dot{s} = f(s, i, p) \tag{1}$$

The measured outputs o from the system can be expressed in terms of the state, input, and parameters through some function h that forms the measurement equations as shown in equation 2.

$$o = h(s, i, p) \tag{2}$$

Functions f and h are nonlinear. However, they can be linearized about an equilibrium point which is a set of states, inputs, and parameters that satisfy the following condition.

$$0 = f(s_{\text{eq}}, i_{\text{eq}}, p_{\text{eq}}) \tag{3}$$

This results in a linear relationship between state, input, output, and the rate of change of the state as shown in equation 4 which is called a state space model

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{4}$$

where x, u, and y are the state, input, and output vector for the linearized system. Matrices A, B, C, and D are found by evaluating the Jacobian of f and h about the equilibrium point.

$$\begin{aligned} A &= \left.\frac{\partial f}{\partial s}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} & B &= \left.\frac{\partial f}{\partial i}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} \\ C &= \left.\frac{\partial h}{\partial s}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} & D &= \left.\frac{\partial h}{\partial i}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} \end{aligned} \tag{5}$$

Equation 4 can be further transformed by defining the input u in terms of x as shown in equation 6

$$\begin{aligned} u &= -Kx \\ \dot{x} &= (A - BK)x \\ y &= Cx + Du \end{aligned} \tag{6}$$

where K is a gain matrix that must be tuned for optimal control.

In addition to the linearized EOM and measurement equations in equation 4, there are also a set of equations relating the input u and outputs y to $\hat{x}$, which is an estimate of the true state x. This estimate is what the controller considers to be the state during flight but is not the true state x because it is not a direct measurement of all the state variables. Even if they were, those measurements themselves would only be an informed "estimate" of the value of the true state. This relationship between $\hat{x}$, u and y is shown in equation 7 where L is a matrix that must be tuned for optimal state estimation.

$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} + Du - y) \tag{7}$$

The kinematic equations for the drone, which partially form the definition of function f, are described in equation 8. $\dot{O}_1^0$ describes the time rate of change of o_x, o_y, and o_z, which are the x, y, z positions of the drone in the global frame (frame 0). $v_{0,1}^1$ describes the velocity of the drone in the drone's body frame (frame 1). Matrix R_1^0 is the rotational matrix that re-expresses vectors from frame 1 to frame 0. $\omega_{0,1}^1$ is the drone's rotational vector expressed in frame 1. ψ, θ, and ϕ are the Euler angles of frame 1 with respect to frame 0 and denotes the orientation of the drone.

$$\dot{O}_1^0 = R_1^0 v_{0,1}^1 \qquad \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = N\omega_{0,1}^1 \tag{8}$$

Equation 8 in its fully expanded form is shown below.

$$\begin{bmatrix} \dot{o_x} \\ \dot{o_y} \\ \dot{o_z} \end{bmatrix} = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi) & \sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi) \\ \sin(\psi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & -\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$
$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \\ 0 & \cos(\phi) & -\sin(\phi) \\ 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{9}$$

B. Linear Quadratic Regulator and Bryson's Rule

In order to optimize for the K and L matrices, the linear quadratic regulator method was used. This involves minimizing the following functions for the controller[5] and observer[6] respectively:

$$\begin{aligned} Controller: \underset{u_{[t_0,\infty)}}{\text{minimize}} \quad & \int_{t_0}^{\infty} \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt \\ \text{subject to} \quad & \dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0. \end{aligned} \tag{10}$$

$$\begin{aligned} Observer: \underset{n_{[t_0,\infty)}\, d_{[t_0,\infty)}}{\text{minimize}} \quad & \int_{t_0}^{\infty} \left(n(t)^T Q n(t) + d(t)^T R d(t) \right) dt \\ \text{subject to} \quad & \dot{x}(t) = Ax(t) + Bu(t) + d(t) \\ & y = Cx(t) + Du(t) + n(t). \end{aligned} \tag{11}$$

For the controller, increasing R would penalize the inputs more and while increasing Q would penalize the error in the states more. For the observer, increasing Q would call for the system to trust the sensors more, whereas increasing R would tell the system to trust the dynamical model more. Bryson's Rule [7] were used to get a baseline for the starting point of the controller R matrix. This involved finding the difference between the drone's maximum motor output and the average motor output required for hover. This value was then adjusted with the force and moment coefficients. The diagonal components of the controller R matrix were populated with the inverse of these values as a baseline and multiplied by a scalar depending on the various flight tests.

C. Controller Implementation

The goal of the controller is to allow the CrazyFlie drone to calculate the right amount of torque and thrust it should be generating based on the difference between the current state and the desired state (which is the basis for the equation $u = -Kx$). The EOM for the states was defined as shown below

$$\dot{s} = \begin{bmatrix} \dot{o_x} \\ \dot{o_y} \\ \dot{o_z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{bmatrix} = \begin{bmatrix} v_x\cos(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)) + v_z(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)) \\ v_x\sin(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi)) + v_z(-\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi)) \\ -v_x\sin(\theta) + v_y\sin(\phi)\cos(\theta) + v_z\cos(\phi)\cos(\theta) \\ \frac{w_y\sin(\phi)}{\cos(\theta)} + \frac{w_z\cos(\phi)}{\cos(\theta)} \\ w_y\cos(\phi) - w_z\sin(\phi) \\ w_x + w_y\sin(\phi)\tan(\theta) + w_z\cos(\phi)\tan(\theta) \\ \frac{gm\sin(\theta) + mv_yw_z - mv_zw_y}{m} \\ \frac{-gm\sin(\phi)\cos(\theta) - mv_xw_z + mv_zw_x}{m} \\ \frac{f_z - gm\cos(\phi)\cos(\theta) + mv_xw_y - mv_yw_x}{m} \\ \frac{J_yw_yw_z - J_zw_yw_z + \tau_x}{J_x} \\ \frac{-J_xw_xw_z + J_zw_xw_z + \tau_y}{J_y} \\ \frac{J_xw_xw_y - J_yw_xw_y + \tau_z}{J_z} \end{bmatrix}$$

where inputs i and relevant parameters p are

$$i = \begin{bmatrix} \tau_x & \tau_y & \tau_z & f_z \end{bmatrix} \qquad p = \begin{bmatrix} m & J_x & J_y & J_z & g \end{bmatrix}$$

The mass of the drone was measured to be 35 grams (0.035 kg). g was assumed to be $9.81 m/s^2$ (which is precise enough for this project). The moments of inertia J were measured by measuring the drone's period of oscillation T when oscillating about a fixed point distance r away from its center of mass. This analysis was performed following the instructions provided for lab 3 for all three axes. Equation 12 shows how the moment of inertia can be solved for some period T and r.

$$J = m\left(gr\left(\frac{T}{2\pi}\right)^2 - r^2\right) \tag{12}$$

After finding all relevant parameters, the EOM were then linearized about this set of states and inputs.

$$s_{eq} = \begin{bmatrix} 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad i_{eq} = \begin{bmatrix} 0 & 0 & 0 & mg \end{bmatrix}$$

This was confirmed to be an equilibrium point because $\dot{s}$ was evaluated to be a zero vector at these values. (It should be noted that the EOM has no dependence on o_z so the choice of 0.5 m in s_{eq} is inconsequential.) Linearizing the EOM expresses the drone's dynamics in the state space model shown in equation 6 where the state x and control input u are

$$x = \begin{bmatrix} o_x \\ o_y \\ o_z - 0.5 \\ \psi \\ \theta \\ \phi \\ v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \qquad u = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ f_z - mg \end{bmatrix} \tag{13}$$

To control the drone's location, reference tracking was used where in which u is defined as follows

$$u = -K(x - x_{\text{des}})$$

where K is the gain matrix that must be tuned for the drone to provide optimal tracking and x_{des} is the desired state of the drone. By definition, u will act on the drone such that $x - x_{\text{des}}$ approaches zero.

LQR was utilized to find and tune the optimal gain matrix. Bryson's rule was used to initially define the R matrix. Q was initially defined as the Q matrix that produced the lowest RMSE values for Shirley and Seongyong's group in Lab 6. Upon iterative tuning, the final Q and R matrices had the following values along their diagonal entries.

$$Q = \begin{bmatrix} 120 & 80 & 75 & 40 & .01 & .01 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 5045605 & 5045605 & 38586096 & 566 \end{bmatrix} \tag{14}$$

Error in the x, y, z positions were penalized most heavily. Yaw (ψ) was also penalized heavily due to the drone's tendency to produce large yaw errors. The penalty on pitch (θ) and roll (ϕ) were decreased because of the high frequency oscillations observed on these states. The remaining penalties were left at its initial value of 1. The resulting gain matrix from the Q and R matrices in equation 14 was

$$K = \begin{bmatrix} 0 & -0.004 & 0 & 0 & 0 & 0.0045 & 0 & -0.002 & 0 & 0.0006 & 0 & 0 \\ 0.0049 & 0 & 0 & 0 & 0.0049 & 0 & 0.0022 & 0 & 0 & 0 & 0.0006 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0003 \\ 0 & 0 & 0.3639 & 0 & 0 & 0 & 0 & 0 & 0.1651 & 0 & 0 & 0 \end{bmatrix}$$

D. Observer Implementation

The goal of the observer is to allow the CrazyFlie drone to generate an estimate for its linearized state ($\hat{x}$) during flight so the controller can use this information to calculate the control inputs (which will be $-K\hat{x}$). The definitions of the state estimate vector $\hat{s}$, input vector i, and output vector o is shown below.

$$o = \begin{bmatrix} lh_x \\ lh_y \\ lh_z \end{bmatrix} \qquad \hat{s} = \begin{bmatrix} o_x \\ o_y \\ o_z \\ \psi \\ \theta \\ \phi \\ v_x \\ v_y \\ v_z \end{bmatrix} \qquad i = \begin{bmatrix} w_x \\ w_y \\ w_z \\ a_z \end{bmatrix}$$

The output vector consists of the x, y, z position data obtained from the lighthouse positioning system (denoted as lh_x, lh_y, and lh_z). The state estimation vector $\hat{s}$ encapsulates the same information as the state vector s except for the angular velocities, which are treated as an input alongside a_z. State estimates for w_x, w_y, and w_z are not necessary because they are directly measured very accurately by the CrazyFlie's onboard IMU with very little noise. With these definitions for states and inputs, the EOM of the system is shown in equation 15.

$$\begin{bmatrix} \dot{o_x} \\ \dot{o_y} \\ \dot{o_z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} v_x \cos(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)) + v_z(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)) \\ v_x \sin(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi)) + v_z(-\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi)) \\ -v_x \sin(\theta) + v_y \sin(\phi)\cos(\theta) + v_z \cos(\phi)\cos(\theta) \\ \frac{w_y \sin(\phi)}{\cos(\theta)} + \frac{w_z \cos(\phi)}{\cos(\theta)} \\ w_y \cos(\phi) - w_z \sin(\phi) \\ w_x + w_y \sin(\phi)\tan(\theta) + w_z \cos(\phi)\tan(\theta) \\ g\sin(\theta) + v_y w_z - v_z w_y \\ -g\sin(\phi)\cos(\theta) - v_x w_z + v_z w_x \\ a_z - g\cos(\phi)\cos(\theta) \end{bmatrix} \tag{15}$$

The measurement equation is shown in equation 16.

$$\begin{bmatrix} lh_x \\ lh_y \\ lh_z \end{bmatrix} = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} \tag{16}$$

Linearizing equations 15 and 16 about s_{eq} and i_{eq}

$$s_{eq} = \begin{bmatrix} 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad i_{eq} = \begin{bmatrix} 0 & 0 & 0 & g \end{bmatrix}$$

gives the state space model for the observer from equation 7 (reproduced below) where the state estimate $\hat{x}$ and input u are

$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} + Du - y)$$

However, state ψ in $\hat{s}$ was not observable due to the observability matrix having a rank of 8. Thus a reduced state vector $\hat{x}_{obs}$ excluding ψ was used. Matrices A and C were also reduced accordingly to obtain A_{obs} and C_{obs}.

$$\hat{x}_{obs} = \begin{bmatrix} o_x \\ o_y \\ o_z - 0.5 \\ \theta \\ \phi \\ v_x \\ v_y \\ v_z \end{bmatrix} \qquad u = \begin{bmatrix} w_x \\ w_y \\ w_z \\ a_z - g \end{bmatrix} \tag{17}$$

The optimal "gain matrix" L was obtained using LQR. The diagonal entries of the Q and R entries were calculated using the square inverse of the standard deviation of the error for the corresponding state and measurement. The error values were obtained by reproducing the steps from lab 7. The drone was instructed to fly in a square using the default observer. The standard deviation for the error in the state estimates and measurements are tabulated below.

State Estimates	o_x	o_y	o_z	ψ	θ	ϕ	v_x	v_y	v_z
Std	0.026	0.026	0.033	0.018	0.103	0.203	0.149	0.115	0.187

Measurements	lh_x	lh_y	lh_z
Std	0.003	0.004	0.002

The standard deviation of the error between the ground truth measurement y and predicted measurement ($C\hat{x} + Du$) during the flight were used to tune Q. The standard deviation of the error between ground truth $\dot{x}$ (obtained via finite difference) and state estimated by the dynamic model ($Ax + Bu$) was used to tune R. It is notable that the error in the position measurements from the lighthouse is very small in comparison to the error in the state estimates made by the dynamic model. The final Q and R matrices had the following values along its diagonal entries.

$$Q = \begin{bmatrix} 111111 & 62500 & 250000 \end{bmatrix} \quad R = \begin{bmatrix} 1479.3 & 1479.3 & 918.3 & 94.3 & 24.3 & 45.0 & 75.6 & 28.6 \end{bmatrix} \tag{18}$$

This resulted in the matrix L shown below

$$L = \begin{bmatrix} 17.76 & 0 & 0 \\ 0 & 17.72 & 0 \\ 0 & 0 & 21.43 \\ 34.33 & 0 & 0 \\ 0 & -50.75 & 0 \\ 120.12 & 0 & 0 \\ 0 & 135.92 & 0 \\ 0 & 0 & 93.5 \end{bmatrix}$$

The 8 observable states were estimated using the state space model for the observer shown below.

$$\dot{\hat{x}} = A_{obs}\hat{x}_{obs} + Bu - L(C_{obs}\hat{x}_{obs} + Du - y)$$

The unobservable ψ was estimated by assuming an initial value of 0 (which is assuming that the drone's x-axis aligns with the global x-axis at takeoff) and by numerically integrating w_z over the course of the flight.

IV. Additional Hardware

The additional hardware used for this project were the LED Ring light and Lighthouse Positioning Deck. As opposed to the optical flow deck, the lighthouse deck allows for the drone to fly in dark or low-light settings. The lighthouse system used infrared signals to communicate between the deck installed on the drone and the two lighthouse base stations. The LED ring light was another component that was used in this drone setup. Although the drone is already outfitted with LEDs, the LED Ring light allows for a more saturated long-exposure picture, as well as multiple options in color combinations.

A. Lighthouse Deck

The lighthouse deck and lighthouse beacons were used for the high-accuracy tracking of the CrazyFlie. The lighthouse system utilizes the angles from the beacons and calibration data to estimate the position of the drone. The beacons provide the sweep angle for all three angles. The lighthouse deck takes in this information and calculated the position of the CrazyFlie. Fig. [8] shows a schematic of how the system is set up. The lighthouse system can use two different methods to estimate the positions. The first one is the crossing beams method, which uses the sweep angles from the beacons and calculates the position of the CrazyFlie. Another method that can be used is to collect the raw sweep angle data directly from the Lighthouse and input it directly into a Kalman filter. For the sake of this project, the lighthouse system was set to use the crossing beams method to ensure that the coordinates could be logged.

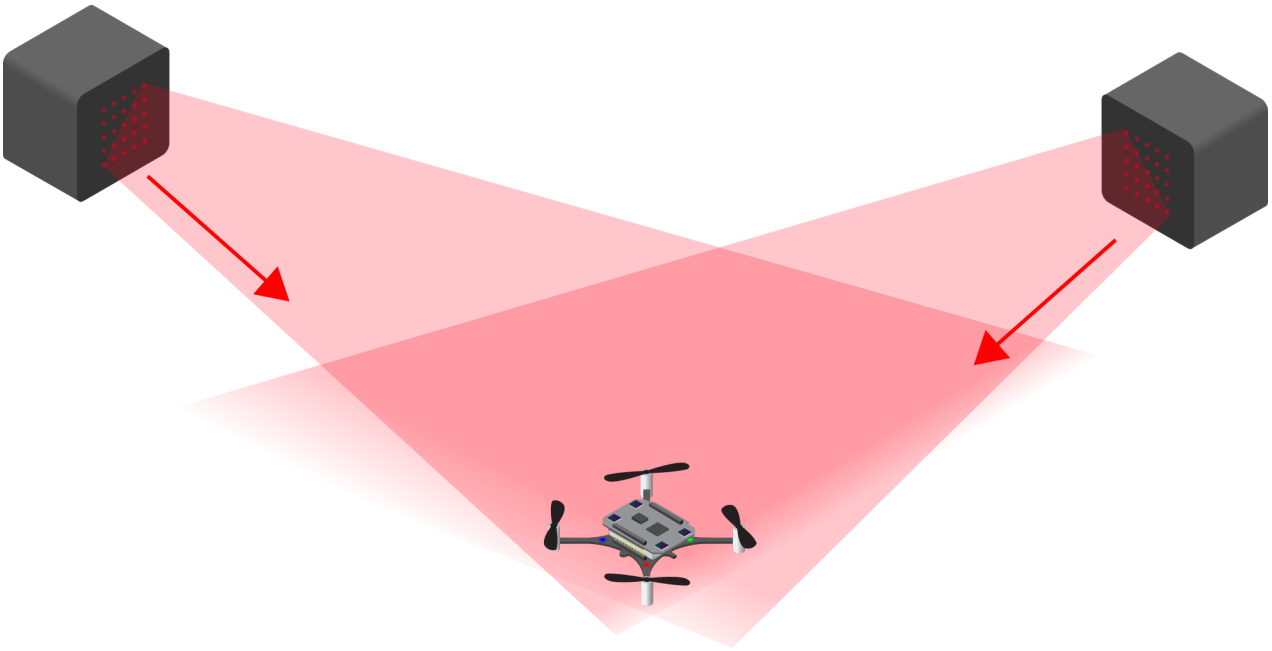

Fig. 1 Overview of Lighthouse Set-Up [8]

V. Implementation

A. Hardware Implementation

The first step in integrating new technologies to the CrazyFlie was to remove the optical flow deck and install the LED Ring on the bottom and lighthouse deck to the top. Fig. 2 and Fig. 3 show the new hardware configuration of the drone. After figuring out how to integrate the new hardware, mass, moments of inertia, and motor force/moment coefficients were found for this new configuration. These are documented in the table below.

$Mass(kg)$	0.0351
$J_x(kg \bullet m^2)$	$1.24 * 10^{-5}$
$J_y(kg \bullet m^2)$	$1.27 * 10^{-5}$
$J_z(kg \bullet m^2)$	$3.32 * 10^{-5}$
k_F	$1.95 * 10^{-6}$
k_M	$7.47 * 10^{-9}$

Once the new coefficients were found, the team began fine-tuning a custom controller. This required changing components of the Q and R matrices

Fig. 2 Lighthouse Desk Installation

Fig. 3 LED Ring Deck Installation

B. Lighthouse Implementation

Another aspect of this project was being able to use the data from the Lighthouse system to enhance the performance of the drone. This required changes to the client code as well as the firmware. The first change was setting the lighthouse method to the crossing beams, which is done by inserting the code seen below in the client code[9].

```
client.cf.param.set_value('lighthouse.method', 0)
```

In order to implement in the firmware, the controller code was modified to accept three floats for the x,y, and z coordinates from the lighthouse deck. A custom function was then written to make sure that these floats were being updated. In addition, the lighthouse_position_est.c file was modified to accept the controller code by adding the controller file as a header in the lighthouse position estimate file.

C. Flight Path Implementation

In order to fulfill the project's purpose of spelling out words, code was implemented to create flight paths that spelled out letters. Twenty-six flight paths were created in total, one for each letter of the alphabet. Since the bright headlights of the lighthouse ring are used for the long exposure photo, the team decided to spell the letters in the drone's x-z plane so that the flight could be filmed from the side to read the letters.

The flight path code followed a distinct formula. All letter flight paths were placed in a function called letter_move, which takes in an input character. The input character is converted to an uppercase letter. A series of if-statements contains code for each letter.

The drone draws the letter within a boundary box with a z-dimension of 0.3 meters and an x-dimension specified by the user. The beginning height of each letter is 0.5 meters off the ground, and the maximum height is 1.0 meters off the ground. A few different variables define x-positions for the drone. For example, the variable x_left is the x-coordinate on the left side of the box, and x_right is the x-coordinate on the right side. These variables are coded to change depending on the current location of the drone with respect to its origin. This code is documented in the team GitHub[9].

At the start of all letters, the headlights are assumed to be off. The drone moves as necessary, turning the headlights on when following a trajectory that is part of the letter and turning them off when moving off the letter trajectory. The drone returns to the lower right corner of the boundary box at the end of each letter and finishes by switching the headlights off if they are not already off. Fig 4 shows a conceptual drawing of what the boundary box and move commands might look like for the name "NOOR."

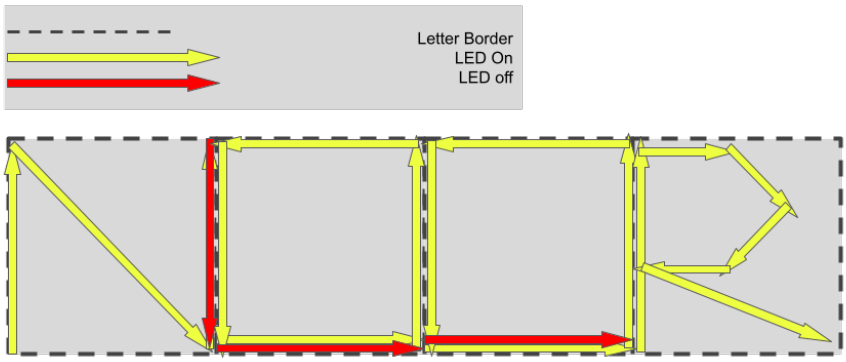

Fig. 4 Flight Path Implementation for NOOR

Outside of the letter_move function, an iterator is used to cycle through all the letters of the input string. It calls letter_move on each letter, moving the drone slightly to the right in the x-direction after each letter to create a small gap between letters. This prevents the lines of neighboring letters from overlapping as they do in Fig. 4.

VI. Results and Discussion

Overall, the team was able to successfully fine-tune a controller and incorporate the lighthouse positioning system into an observer. Fig. 5 shows the long-exposure photo of the drone's flight with both a custom controller and observer implemented. The drone legibly spells out "NOOR" as intended. The drone's recorded position as observed by the custom and the default observer is also plotted in Fig. 6 and Fig. 7, which shows the drone's trajectory in the x-y plane and the x-z plane. The RMSE between the design position and the observed position is shown in Table 1, and the RMSE between the custom observer and the default observer is shown in Table 2 to show how much the drone varied from the design path and to show how different the custom observer was compared to the default observer. Fig. 7 shows that the implementation of the custom controller and observer still results in a somewhat shaky trajectory in the x-z plane. However, these errors are not due to the observer, but more likely the controller. This is shown by RMSE results from Table 2. Here the errors are extremely small and not surpassing 0.02 as a value. This shows that the default observer and the custom observer are not too far off. Looking at the RMSE results from Table 1, the error between the design position and the observed position, there was no RMSE value that was less than 0.05. Coupled with the fact that the observer was just fine, the RMSE of the design position can be blamed on the controller.

Specifically, the drone tended to drift to the left as seen in Fig. 7. The best way to fix this error is to further tune the controller. When creating the controller for this project, it was tested for its ability to reliably complete a flight in spelling out the letters. For future development, however, the controller could be developed with a set of strict RMSE requirements to satisfy.

The team also looked at how far the drone drifted in the y-direction to gauge the success of the project so the EOH team can have solid information on how safe the project is. Ideally, the drone should not have drifted at all, and the top-down view should be just a straight line along the 0.0 y-position in Fig 6. The black line shows a 0.2 m drift in either direction. The only time it goes above this limit is during takeoff, so as mentioned before, the best way to fix this is to fix the controller for a smoother takeoff to fit the team's safety parameters.

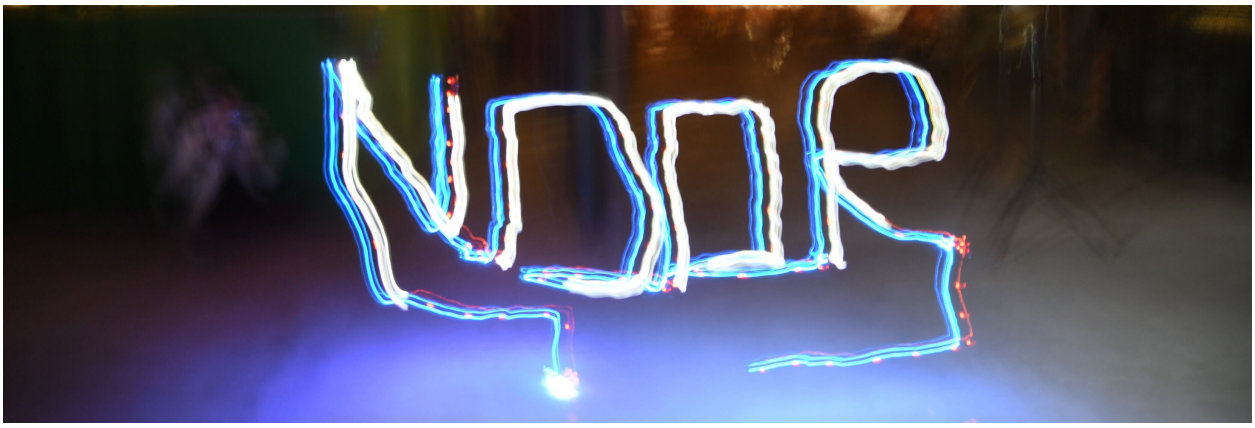

Fig. 5 Photo Taken of Drone's Path

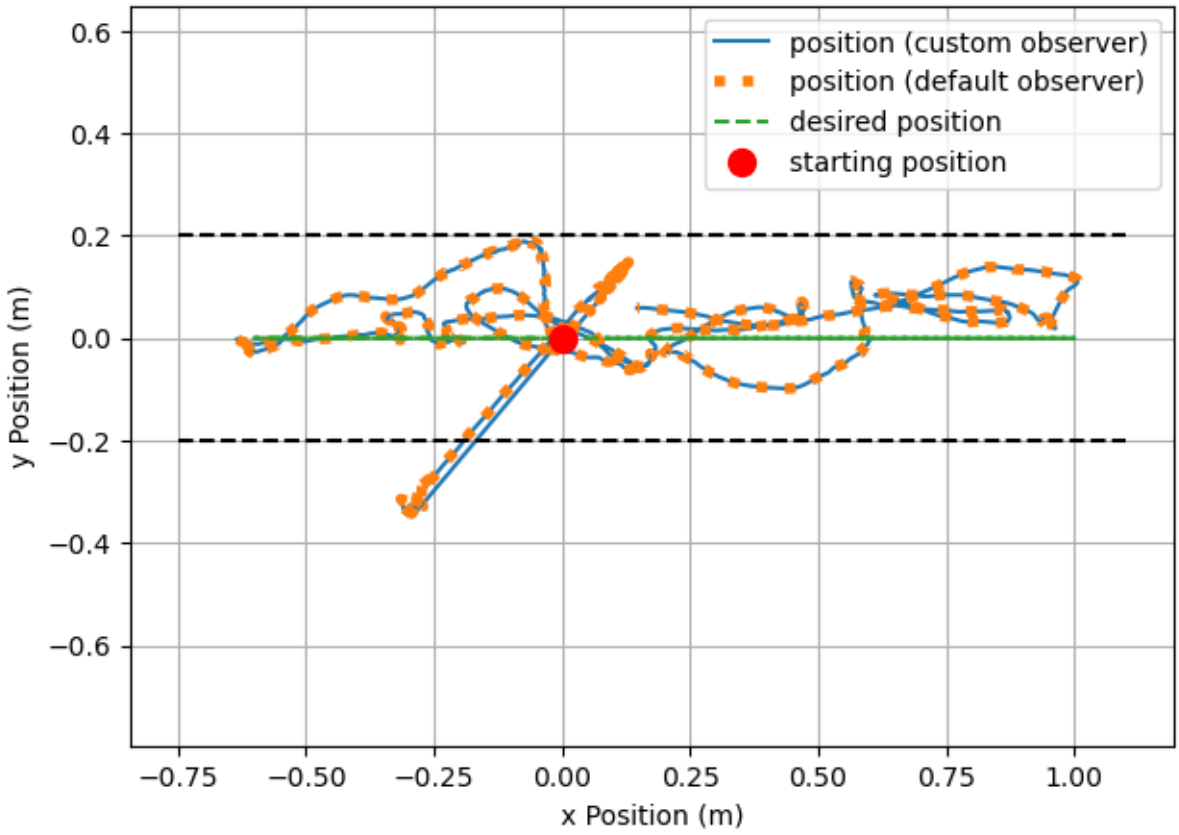

Fig. 6 Drone Flight Path on the x-y Plane

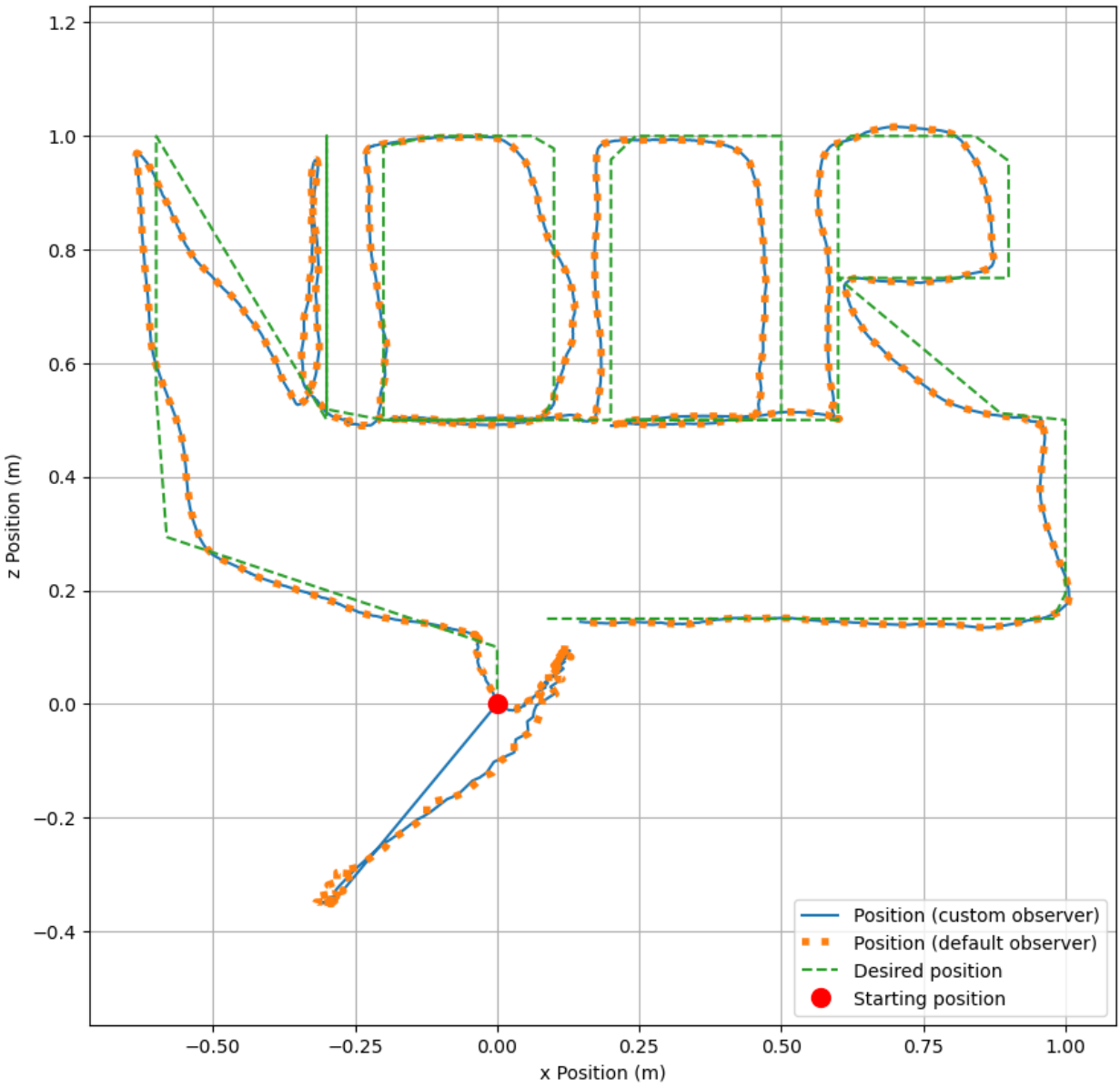

Fig. 7 Drone Flight Path on the x-z Plane

Table 1 RMSE of Design Position

x	y	z
0.083	0.058	0.068

Table 2 RMSE of Observer Position and Orientation

x	y	z	ψ	θ	ϕ
0.012	0.013	0.014	0.003	0.004	0.003

VII. Future Improvements

Though the team accomplished its goals, there is room for further improvement. Given more time, the team would implement a function to automatically create flight paths based on inputted shapes, such as typed or handwritten letters. Since the aim of the project was to create a result that could be exhibited at EOH, this would allow visitors to write their names on a tablet in order to dictate the drone's flight path. Further improvements to the controller could also be made to increase the ability of the drone to fly very accurately in very tight spaces. This would allow the drone to make smaller letters, allowing for longer words or names. The team would also experiment with adding another Lighthouse beacon and spreading the beacons out further to increase the size of the canvas the drone can fly within. While the final long-exposure pictures turned out well, there can still be more to improve the quality of the paintings. One flaw of the current photos are that the LEDs don't turn off fast enough which results in flight paths that don't require the LEDs to be highlighted. This includes the flight path moving from one letter to the next letter and the takeoff and landing.

VIII. Conclusion

Although a formal list of requirements was not provided for this project, the project was considered a success due to the fact that a custom controller and custom observer were able to create a legible long exposure photo. With further fine-tuning, this project would be ready to participate in outreach events. Although it most likely will not be ready for EOH applications which are due Dec 18th, 2022, it will be ready for the annual Girls Learning Aeronautical and Space Sciences Day hosted by Women in Aerospace. Continuing this project will involve further fine-tuning the controller and observer as well as validating that all 26 letters can be written. With the additional time, the team would also like to ensure that names are drawn more clearly. To accomplish this, not only will the team ensure that the controller and observer are resulting in low errors, but make sure that the final setup is ideal. This includes making the surrounding area as dark as possible by putting up black drapes around the canvas area.

Acknowledgments

The authors would like to thank Professor Tim Bretl and teaching assistant Tiger Hou for enduring progress and guiding us through this project. The team would also like to thank Dan Block for letting us use his lab space for testing purposes. The team would also to thank Eric Monson, Anna Hylbert, Matt Taylor, Caleb Carrigan for letting us use the lighthouse beacons. The team also thanks David Robbins for providing C code knowledge.

References

[1] Bitcraze, "The Lighthouse positioning system," 2022. URL https://www.bitcraze.io/products/led-ring-deck/.

[2] Yang, A. N. H., Daniel; Madrzyk, "Long Exposure Photography with Crazyflie LED's," 2021.

[3] Ahmed, L. M. E. M. A., Mujahid; English, "AE 483 Project: Long Exposure Light Tracing of a Fully-Observable Drone System," 2021.

[4] Bitcraze, "Light painting with Crazyflie and the Lighthouse deck," , 2019. URL https://www.bitcraze.io/2019/03/lighthouse-painting/.

[5] Bretl, T., "Controller Design," Retrived From Canvas, 2022.

[6] Bretl, T., "Observer Design," Retrived From Canvas, 2022.

[7] Bretl, T., "Bryson's Rule," Retrived From Canvas, 2022.

[8] Bitcraze, "Lighthouse Positioning System," , 2019. URL https://www.bitcraze.io/documentation/system/positioning/ligthouse-positioning-system/.

[9] Hong, S., "Github," , 2019. URL https://github.com/Seongyong36/ae483-final-project.

Appendix

A. Appendix A

Day	Person	Task
Nov 2	Noor	Wrote Milestones
Nov 2	Audrey	Revised/wrote Abstract
Nov 2	Shirley	Began writing introduction, researched bitcraze components
Nov 2	Seongyong	Created GitHub Account for project and Bill of Materials (BOM)
Nov 4	Noor	Wrote the introduction
Nov 4	Audrey	Wrote the introduction
Nov 4	Shirley	Contributed to the introduction and milestones, helped generate references
Nov 4	Seongyong	Wrote the introduction
Nov 8	Noor, Shirley, Seongyong, Audrey	Created presentation for progress meeting
Nov 9	Noor, Shirley, Seongyong, Audrey	Attended progress meeting
Nov 10	Seongyong	Reached out to Eric Monson. Acquired permission to use his personal SteamVR Base Station as beacons for the lighthouse system.
Nov 18	Shirley	Attempted to integrate LED ring and lighthouse board to CrazyFlie.
Nov 21	Seongyong	Attempted to reattach broken headers on LED ring. Attempted to rewire headers to allow LED ring to fit above the CrazyFlie's battery. Attempt was unsuccessful and a new LED ring was ordered.
Nov 23	Audrey	Wrote code for letters A-M.
Nov 27	Noor	Flew drone to draw out Audrey's letters.
Nov 28	Noor, Audrey, Seongyong	Held group progress meeting. Finalized the decision to use lighthouse system in place of the optical flow deck.
Nov 28	Seongyong	Learned how to set up and calibrate the lighthouse beacons.
Nov 30	Seongyong, Shirley, Audrey	Practiced lighthouse beacon setup and calibration. Discovered that loco positioning system interferes with Lighthouse beacons.
Dec 2	Audrey	Finalized move commands to spell letters.
Dec 2	Noor	Started outlining project video and presentation
Dec 2	Shirley, Seongyong	Measured moments of inertia of drone by utilizing code given for lab 3.
Dec 3	Noor	Continued outlining and drafting project video and presentation.
Dec 3	Seongyong, Shirley	Measured motor coefficients (re-did lab 4). Tuned and implemented controller in firmware (re-did lab 6). Spelled NO using custom controller and default observer. Took first long exposure photo of flight.
Dec 4	Shirley, Seongyong, Audrey, Noor	Investigated ways to allow `controller_ae483.c` to access and log lighthouse position measurements.

Dec 5	Shirley, Noor, Seongyong	Investigated ways to allow `controller_ae483.c` to access and log lighthouse position measurements.
Dec 5	Seongyong	Implemented changes in firmware to allow `controller_ae483.c` to access and log lighthouse position measurements made in `lighthouse_position_est.c`.
Dec 5	Seongyong, Noor	Re-derived measurement equations. Measured RMSE of the states from the dynamic model and lighthouse position measurements (re-did lab 7). Calculated L matrix based on RMSE (re-did lab 8).
Dec 5	Seongyong, Audrey	Implemented observer in firmware and Jupyter notebook. Demonstrated that the drone can fly to spell NOOR (re-did lab 9).
Dec 6	Seongyong	Crashed drone. Replaced motors 1 and 3.
Dec 6	Seongyong, Noor, Shirley	Took long-exposure photo and video of flight for project video.
Dec 6	Noor	Produced Project Video.
Dec 10	Seongyong	Worked on drafting the Theory section.
Dec 10	Audrey	Wrote the flight path implementation section. Wrote Future Improvements section.
Dec 14	Noor	Cleaned up Jupyter notebook files for submission.
Dec 14	Noor	Wrote some Results and Discussion
Dec 14	Audrey	Edited introduction, flight path implementation sections. Cleaned up Jupyter notebook files for submission. Proofread report.