Inverted Pendulum on Crazyflie Quadcopter

Anshuk Chigullapalli, Isabella Watters, Rishi Patel, and Harry Zhao *
University of Illinois at Urbana-Champaign, Urbana, IL, 61820

The goal of this project is to create a controller capable of balancing an inverted pendulum on the Crazyflie quadcopter. This is implemented by utilizing the provided sensor decks and the Motion Capture (MoCap) system present in the lab environment. A pendulum is mounted atop the drone with a hinge that allows for one angular degree of freedom. The equations of motion for the pendulum-quadcopter system are derived and linearized about the equilibrium point where the pendulum is in an upright position. The pose of the pendulum is estimated using the MoCap system and tracking markers on both the pendulum and drone. This pose data is then incorporated into the system observer. During testing, the angular error of the pendulum is analyzed to quantify error in the controller and observer performance. Time permitting, the system will be made independent of the MoCap system.

I. Nomenclature

α	=	angle of the pendulum with respect to the world frame, radians
θ	=	pitch of the drone with respect to the world frame, radians
ψ	=	yaw of the drone with respect to the world frame, radians
ϕ	=	roll of the drone with respect to the world frame, radians
x_1	=	x position of reference point with respect to the world frame, meters
x_2	=	x position of pendulum tip with respect to the world frame, meters
x_3	=	x position of pendulum base with respect to the world frame, meters
y_1	=	y position of reference point with respect to the world frame, meters
y_2	=	y position of pendulum tip with respect to the world frame, meters
y_3	=	y position of pendulum base with respect to the world frame, meters
z_1	=	z position of reference point with respect to the world frame, meters
z_2	=	z position of pendulum tip with respect to the world frame, meters
z_3	=	z position of pendulum base with respect to the world frame, meters
m	=	mass of drone, grams
m_{pen}	=	mass of pendulum, grams
l_{pen}	=	length of pendulum, meters
g	=	acceleration due to gravity, m/s^2
f_z	=	thrust of drone in z-direction, lbf
τ_x	=	torque of drone in x-direction
τ_y	=	torque of drone in y-direction
τ_z	=	torque of drone in z-direction
o_x	=	position of drone in x-direction, meters
o_y	=	position of drone in y-direction, meters
o_z	=	position of drone in z-direction, meters
v_x	=	velocity of drone in x-direction, m/s
v_y	=	velocity of drone in y-direction, m/s
v_z	=	velocity of drone in z-direction, m/s
w_x	=	angular velocity of drone about x-direction, rad/s
w_y	=	angular velocity of drone about y-direction, rad/s
w_z	=	angular velocity of drone about z-direction, rad/s
a_x	=	angular velocity of drone in x-direction, radians/s
a_y	=	angular velocity of drone in y-direction, radians/s

*Student, Department of Aerospace Engineering, Talbot Laboratory 104 S. Wright St, 61820

a_z = angular velocity of drone in z-direction, radians/s

II. Introduction

The inverted pendulum is a classical control problem. With its center of mass above its pivot point, it presents an inherently unstable and nonlinear system. In order to keep the pendulum balanced, a feedback control system works to monitor the angle of the pendulum and restore the pivot point back under the center of mass.

The inverted pendulum can be seen in many real world applications. The human body serves as one example of such. With its center of mass above the pivot point at the feet, the body must generate small, constant muscular corrections to keep from falling over [1]. Segway vehicles specifically harness the unstable nature of inverted pendulums for transportation. When a rider leans forward and perturbs the pendulum from the upright, equilibrium position, the vehicle must travel forward in an attempt to re-stabilize. Perhaps one of the most applicable instances of inverted pendulums is found in rockets. Model rocket stability can be modeled as an inverted pendulum with lift and drag acting as the restorative forces [2]. SpaceX's vertically landing Falcon Heavy boosters are another famous rocketry application of inverted pendulums.

The system used for this project consists of a custom pendulum mounted on a Crazyflie quadcopter. The pendulum, comprised of thin metal tubing and a point mass, is attached to the drone using a hinge. This constrains the pendulum in all but one angular degree of freedom. Meanwhile, the quadcopter is unconstrained, offers exceptional agility, and is capable of large/quick translational accelerations [3]. In order to incorporate the pendulum into the system, the OptiTrack Motion Capture (MoCap) system is utilized in the control system's observer. The MoCap system consists of a network of motion tracking cameras positioned around the testing area. Small reflective orbs attached to the pendulum are tracked by the MoCap system. The pose of the pendulum is then reported with an error of less than 0.3 mm [4]. Accounting for a delay between the interfacing systems, this pose data is used by the controller to provide real-time motor commands to the quadcopter. A detailed description of the MoCap observer implementation is found in later sections.

In an effort to simplify the implementation of the flying inverted pendulum, the "swing up" maneuver at the beginning of flight, involving moving the pendulum from resting on the ground to vertical above the drone, will not be controlled for. Instead, the pendulum will be held in the upright position until the drone flies to its initial hover altitude, and then gently released.

Numerous previous projects have made use of the MoCap system to successfully track and inform the flight of the drone [5]. Additionally, the inverted pendulum problem is not new and has been solved on numerous occasions through various applications, including mounted on a quadcopter. Through this project, the accuracy of the motion capture system is combined with the agility of a a miniaturized quadcopter such as the Crazyflie, lending to a unique implementation of the problem. Due to the accessibility of the Crazyflie quadcopter and simplicity of pendulum design, this system can also be used in educational outreach applications. By introducing students to the complex idea of aerospace control systems at a young age using the drone-inverted pendulum system, they can begin to develop their "engineering intuition" and explore diverse interests.

This report will first detail the theory and dynamic model. It will then describe the physical drone-pendulum system and the implementation of the controller and observer, both in simulation and hardware. Finally, experimental results will be presented and discussed.

III. Dynamics

The dynamics of the joint drone-pendulum system were explored in two steps. At first, the dynamics of the pendulum were derived in a planar problem that only considered the motion in the x and z axes of the world frame, the pitch of the drone θ and the angle of the pendulum with the world z axis, represented by α. Next, the full drone system's dynamics are considered and the linearized equations for the entire pendulum-drone system are derived. These full equations are used to generate the controllers used in this project.

A. Inverted Pendulum Dynamics in the X-Z Plane

The inverted pendulum's dynamics are considered in the X-Z plane, as sketched in Figure 1.

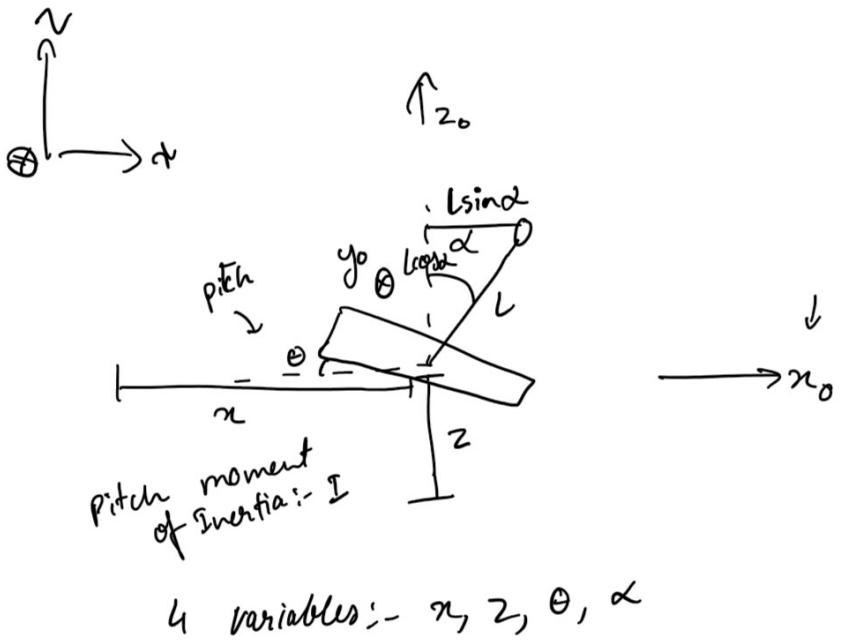


Fig. 1 Planar Problem to Derive the Pendulum's Dynamics

This was because the pendulum's hinge only acts in one direction, and so the planar treatment of the dynamics fully conveys the motion of the pendulum with respect to the pendulum. This system was solved by using Lagrangian equations and x, z, θ and α as the generalized co-ordinates. By calculating the kinetic energy and the potential energy of the full system in this plane, the expression for the Lagrangian was derived and is presented in Equation 1.

$$\mathcal{L} = \frac{J_y\dot{\theta}^2}{2} + \frac{m\left(\dot{x}^2+\dot{z}^2\right)}{2} + \frac{m_{pen}}{2}\left(l_{pen}^2\dot{\alpha}^2 + 2l_{pen}\dot{\alpha}\left(-\sin(\alpha)\dot{z}+\cos(\alpha)\dot{x}\right)+\dot{x}^2+\dot{z}^2\right) - gmz - gm_{pen}\left(l_{pen}\cos(\alpha)+z\right) \tag{1}$$

The Lagrange equations are taken with respect to the generalized co-ordinates as shown in Equation 2. The generalized forces are considered when deriving the equations of motion.

$$\begin{aligned}
\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{\partial\mathcal{L}}{\partial x} &= f_z sin(\theta)\\
\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{z}} - \frac{\partial\mathcal{L}}{\partial z} &= f_z cos(\theta)\\
\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{\theta}} - \frac{\partial\mathcal{L}}{\partial\theta} &= \tau_y\\
\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{\alpha}} - \frac{\partial\mathcal{L}}{\partial\alpha} &= 0
\end{aligned} \tag{2}$$

Here, f_z and τ_y are the thrust and pitch torque generated by the drone respectively. The Lagrange equations above give four equations of motion, which were then solved to derive the expressions for $\ddot{x}$, $\ddot{z}$, $\ddot{\theta}$, and $\ddot{\alpha}$ in the planar problem. These equations are given in Equations 3, 4, 5, and 6.

$$\ddot{x} = \frac{-f_z m_{pen}\sin(\alpha-\theta)\cos(\alpha) + m\left(\dot{\alpha}^2 l_{pen}m_{pen}\sin(\alpha) + f_z\sin(\theta)\right)}{m\left(m+m_{pen}\right)} \tag{3}$$

$$\ddot{z} = \frac{f_z m_{pen}\sin(\alpha)\sin(\alpha-\theta) + m\left(\dot{\alpha}^2 l_{pen}m_{pen}\cos(\alpha) + f_z\cos(\theta) - gm - gm_{pen}\right)}{m\left(m+m_{pen}\right)} \tag{4}$$

$$\ddot{\theta} = \frac{\tau_y}{J_y} \tag{5}$$

$$\ddot{\alpha} = \frac{f_z \sin(\alpha - \theta)}{l_{pen} m} \tag{6}$$

These expressions showcase the non-linearity of the control problem. To evaluate whether this system is controllable, the equations above were linearized into a state-space model. A standard state-space model (not considering the observer) is shown in Equation 7. The four generalized co-ordinates mentioned above and their time derivatives composed the state, and the two inputs were τ_y and f_z.

$$\dot{x} = Ax + Bu \tag{7}$$

$$x = \begin{bmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \\ \alpha \\ \dot{\alpha} \end{bmatrix}, \; u = \begin{bmatrix} \tau_y \\ f_z \end{bmatrix} \tag{8}$$

This state-space model was linearized about the point where the pendulum is in the upright position, i.e. $x = [0]_{8x1}$. The resulting A and B matrices are shown in Equation 9 because they clearly convey some changes to the dynamics of the system.

$$A = \begin{bmatrix} 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 11.1477 & 0 & -1.337 & 0 \\ 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & -74.318 & 0 & 74.318 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 20.0 \\ 0 & 0 \\ 62500.0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{9}$$

From A, it is clear that the new dynamics introduced include $\dot{x}$ being dependent on the angle of the pendulum, and the angular velocity of the pendulum depending on the pendulum angle and the pitch of the drone. To verify whether the above system is controllable, the controllability matrix was calculated using the above A and B matrices. This controllability matrix had a rank of 8, proving that this system was controllable.

Although this planar treatment of the pendulum-drone system was useful in understand the effects of the pendulum on the drone's dynamics and vice versa, it cannot be used to generate a useful controller for the true, three-dimensional world. The following section details the dynamics of the full system and how the dynamics of the pendulum are considered in this case.

B. Dynamics of the Full System

Unlike the abridged state used in the above section, the full system state includes the drone's position in the world-frame, the drone's orientation, the drone's velocity in the body frame, the drone's angular velocity in the body frame, and the pendulum's angular position and angular velocity in the body frame. The inputs to the state include the drone torques about the body x, y, and z axes and the thrust of the drone.

The dynamics of the drone involve considering both the world frame and the body frame. So, it is important to consider the rotation between these frames. This rotation can be represented by the rotation matrix R shown in Equation 10.

$$R = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi) & \sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi) \\ \sin(\psi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & -\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{10}$$

Transforming a body-frame vector with R gives the world-frame result. Similarly, multiplying with R^T converts a vector from the world-frame to the body frame. Similarly, the matrix N is used to transform the body rates w of the drone to the derivatives of the drone angles ψ, θ and ϕ. This is given in Equation 11

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = N w_{0,1}^{1}$$
$$N = \begin{bmatrix} 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \\ 0 & \cos(\phi) & -\sin(\phi) \\ 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \end{bmatrix} \tag{11}$$

Now, the forces of the drone are considered. Here, the team should have incorporated the new equations of motion that were derived in the planar problem in Section III.A. However, due to the planar equations for $\ddot{o}_x$ and $\ddot{o}_z$ being in the world-frame, the conversion to equations that can be used for v_x and v_z was difficult and was not completed in time. As such, the team decided to continue using the expression for the forces as shown in Equation 12, which ignore the effects of the pendulum. Here, the forces in the body frame are represented by f_1.

$$f_1 = R^T \begin{bmatrix} 0 \\ 0 \\ -gm \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} \tag{12}$$
$$= \begin{bmatrix} gm\sin(\theta) \\ -gm\sin(\phi)\cos(\theta) \\ f_z - gm\cos(\phi)\cos(\theta) \end{bmatrix} \tag{13}$$

The torques τ_x, τ_y, and τ_z do not have to be manipulated as they are in the body frame.

Now, all these equations are put together to express the full equations of motion of the system. Equation 14 showcases the state and input of this system.

$$\begin{aligned} x &= [o_x, o_y, o_z, \psi, \theta, \phi, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \alpha, \dot{\alpha}]^T \\ u &= [\tau_x, \tau_y, \tau_z, f_z]^T \end{aligned} \tag{14}$$

It is important to note that the equations of motion for $\ddot{\alpha}$ are taken from Equation 6. This is the primary new addition to the equations of motion for this project.

Now, all these equations are put together to get the nonlinear derivative of the state x shown in Equation 14, and is represented by f in Equation 15.

$$
f = \begin{bmatrix}
v_x \cos(\psi)\cos(\theta) + v_y\left(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)\right) + v_z\left(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)\right) \\
v_x \sin(\psi)\cos(\theta) + v_y\left(\sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi)\right) + v_z\left(-\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi)\right) \\
-v_x \sin(\theta) + v_y \sin(\phi)\cos(\theta) + v_z \cos(\phi)\cos(\theta) \\
\frac{w_y \sin(\phi)}{\cos(\theta)} + \frac{w_z \cos(\phi)}{\cos(\theta)} \\
w_y \cos(\phi) - w_z \sin(\phi) \\
w_x + w_y \sin(\phi)\tan(\theta) + w_z \cos(\phi)\tan(\theta) \\
\frac{gm\sin(\theta) + mv_y w_z - mv_z w_y}{m} \\
\frac{-gm\sin(\phi)\cos(\theta) - mv_x w_z + mv_z w_x}{m} \\
\frac{f_z - gm\cos(\phi)\cos(\theta) + mv_x w_y - mv_y w_x}{m} \\
\frac{J_y w_y w_z - J_z w_y w_z + \tau_x}{J_x} \\
\frac{-J_x w_x w_z + J_z w_x w_z + \tau_y}{J_y} \\
\frac{J_x w_x w_y - J_y w_x w_y + \tau_z}{J_z} \\
\dot{\alpha} \\
\frac{f_z \sin(\alpha - \theta)}{l_{pen} m}
\end{bmatrix} \tag{15}
$$

f is linearized about the point where the pendulum is upright, and the drone has zero angular or translational velocity and is at its desired position. By taking the Jacobian of f with respect to the state x and the input u, the state-space matrices A and B, respectively, are found (the state-space representation is shown in Equation 7). For conciseness, these matrices are not shown in the report but can be viewed in the code.

With the derived A and B, the controllability matrix was found for the system. It was found that this controllability matrix was not full rank, indicating that some elements were not controllable. It is possible that there was an issue with any modifications to the planar problem equations that needed to be made before the equations were added to the full system. The consequences of this are discussed in the following section.

IV. Controller

Over the course of this project, many approaches were taken to design a controller that would be capable of both flying the drone and balancing the inverted pendulum. Although none of them worked particularly well, it was worth exploring the different approaches possible and the affects of the controllers.

All controllers worked on the principle of feedback control, i.e. the input to the system is proportional to the state estimate or the state itself. This is represented by Equation 16.

$$u = -K\hat{x} \tag{16}$$

The difference in design approaches came in the development of K, the controller gain matrix.

A. Direct Pendulum Angle Feedback

Prior to the development of accurate equations of motion for the joint pendulum-drone system, the team attempted to create a controller based on just the drone system (12 states) but adding positive feedback of the pendulum angle to the pitch torque τ_y. The controller can be written in the form of a matrix feedback gain K as shown below in Equation 17.

$$K = \begin{bmatrix} 0 & -0.00264 & 0 & 0 & 0 & 0.00667 & 0 & -0.00210 & 0 & 0.0011 \\ 0 & 0 & 0 & 0 & & & & & & \\ 0.00224 & 0 & 0 & 0.00655 & 0 & 0.00195 & 0 & 0 & 0 & 0.00109 \\ 0 & 0 & 0.05 & 0 & & & & & & \\ 0 & 0 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.00103 & 0 & 0 & & & & & & \\ 0 & 0 & 0.215 & 0 & 0 & 0 & 0 & 0 & 0.196 & 0 \\ 0 & 0 & 0 & 0 & & & & & & \end{bmatrix} \tag{17}$$

This controller, surprisingly, provided oscillating but bounded control of the pendulum angle in simulation when starting at a 0 initial angle. However, when tested on the real system, the team learnt that the control of o_x and v_x was acting against what would be required to control the pendulum. The team termed this the "pull-back" action that the drone was having. Even though elements of the K shown in Equation 17 were tuned, this behavior persisted. The team realized that although the drone was responding in the correct direction, this type of controller would not work.

B. LQR Controller Design

Next, the team used the linearized equations of motion that include the state of the pendulum derived in Section III and used the Linear Quadradic Regulator (LQR) method to design an optimal controller that is capable of controlling the pendulum.

The LQR method gives a feedback controller K based on the optimal control problem shown in Equation 18. To design a controller, the weights on the state error (represented by Q) and the input effort (represented by R) must be tuned to match the priorities of the system goal.

$$\underset{u_{[t_0,\infty)}}{\text{minimize}} \qquad \int_{t_0}^{\infty} \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt \tag{18}$$

$$\text{subject to} \qquad \dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0. \tag{19}$$

The team developed many iterations of this controller by tuning the values of Q and R. These controllers were tested in simulation and on the real system. The final iteration of this controller that was tested multiple times in real-life and is analyzed in Section VIII. The Q and R values are provided in Equation 20 and the final K value is shown in Equation 21.

$$\begin{aligned} Q &= diag([0.01, 7., 5., 1., 20., 0.01, 0.02, 0.8, 1., 1., 0.01, 1., 50., 2.]) \\ R &= diag(\left[10^6, 10^9, 10^6, 10\right]) \end{aligned} \tag{20}$$

$$K = \begin{bmatrix} 0 & -0.002646 & 0 & 0 & 0 & 0.006665 & 0 & -0.002097 & 0 & 0.0011 \\ 0 & 0 & 0 & 0 & & & & & & \\ -3.0 \cdot 10^{-6} & 0 & 0 & 0 & 0.004033 & 0 & -1.6 \cdot 10^{-5} & 0 & 0 & 0 \\ 0.000359 & 0 & -0.004384 & -0.000543 & & & & & & \\ 0 & 0 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.00102 & 0 & 0 & & & & & & \\ 0 & 0 & 0.707107 & 0 & 0 & 0 & 0 & 0 & 0.402772 & 0 \\ 0 & 0 & 0 & 0 & & & & & & \end{bmatrix} \tag{21}$$

The row of interest is the one that corresponds to the τ_y input. This input controls the movement in the x direction, the pitch of the drone and the corresponding velocities. Thus, it is the primary input that affects the pendulum's angle. Surprisingly, the signs of the gains in this row that correspond to o_x and v_x have switched signs from positive in Equation 17 to negative in Equation 21. This behavior was the fix to the "pull-back" behavior that the team

noticed with the previous method of controller design. It also plays into the sometimes counter-intuitive behavior of an inverted pendulum system. This is explored in further detail in Section VIII.C.

The results of the physical tests of this controller are provided in Section VIII.

V. Observer

The controller requires the state of both the drone and the pendulum in order to implement state-feedback control. The observer converts the sensor outputs into the state needed which includes the full state described in 14. This includes the angle and angular velocity of the pendulum with respect to the world frame. The OptiTrack Motion Capture system was used to observe the state of the pendulum, as is detailed further in Section VII.C. Similar to the controller, the observer was developed using an LQR method which was completed in AE 483 Lab 9.

VI. Simulation

The controller is tested in simulation prior to implementation in hardware. PyBullet, the Python wrapper for the Bullet physics engine, is used to simulate the drone-pendulum system. The simulation used in this paper is a modification of the simulation used in the AE 483 Autonomous Systems Lab course and incorporates major components of this simulation.

A. Model

The drone-pendulum system is modeled using Unified Robot Description Format (URDF). The basic drone model from the AE 483 course is used a basis for the new URDF model. While the basic drone model uses STL files to represent its geometry, the pendulum is modeled parametrically using basic shapes available with URDF. The pendulum rod is represented as a thin cylinder, while the motion capture markers are represented as spheres. The axle allowing the pendulum to freely rotate around one axis is modeled as a continuous joint. Collision behavior is specified for each object so that the pendulum's rotation limits are set by the body of the drone, just like in the real world. Where possible, the dimensions and masses for each object are measured in the real world, and approximate moments of inertia are calculated. These values are used as input in the URDF file to provide physical characteristics which the Bullet physics engine requires to simulate their behavior. It was important that the appropriate measurements and values were used so that the simulation could be considered approximately representative of the system's real-world behavior.

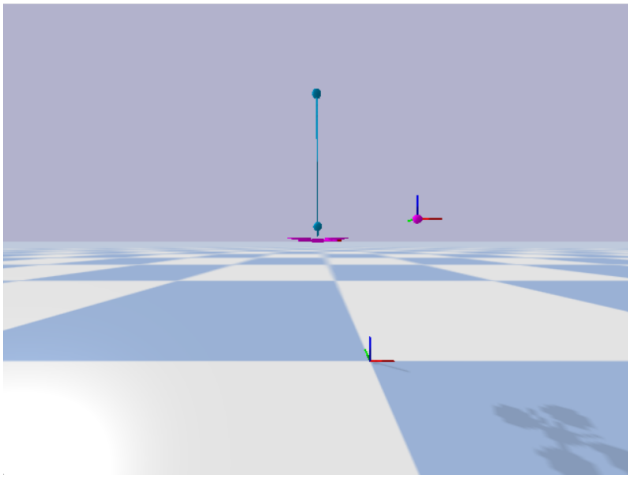


Fig. 2 Pendulum-drone system in PyBullet

B. Simulator

The simulator is also derived from the simulator used in the AE 483 course. The new simulator adds the capability to recognize the pendulum on the drone. Specifically, it obtains the angle between the pendulum and the +z vector

of the drone. This is done by obtaining the joint state of the pendulum joint as specified in the URDF file, which matches the angle definition in the dynamical model of the system derived previously. The angle is added to the values passed to the simulated controller to create a modified state which is compatible with the controller designed in this paper.

As in the course, the simulator is composed of a Python script and a Jupyter Notebook. The Python script contains the Simulator class and forms the backend of the simulation. The Jupyter Notebook is used to create an instance of the Simulator and forms the frontend or user-facing portion of the simulation. The Jupyter Notebook also enables control over the simulation, such as controlling the view or adding and removing drones from the simulation environment. Other features carried over from the course include the capability to log data and record videos.

The controller is implemented in the simulator in a manner similar to hardware. Force and torque commands are calculated, and these are then translated into motor commands. The major difference between the hardware and simulation implementations is that the hardware uses C++ code, while the same lines to generate force, torque, and motor command values are written in Python for the simulation.

In simulation, the controller is able to stabilize the pendulum over a variety of initial conditions, including with the pendulum starting angle offset from vertical. The behavior appears to quite steady–significantly steadier than in the real world, indicating that the simulation is not a completely accurate representation of real-world performance. Discussion of results from the simulation is provided in the Results section. However, the simulation is a useful tool for verifying the theoretical behavior of a designed controller and for tuning of LQR weights.

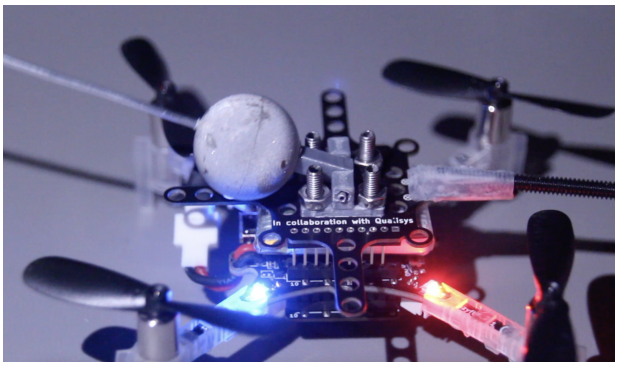
VII. Hardware

A. Crazyflie Drone

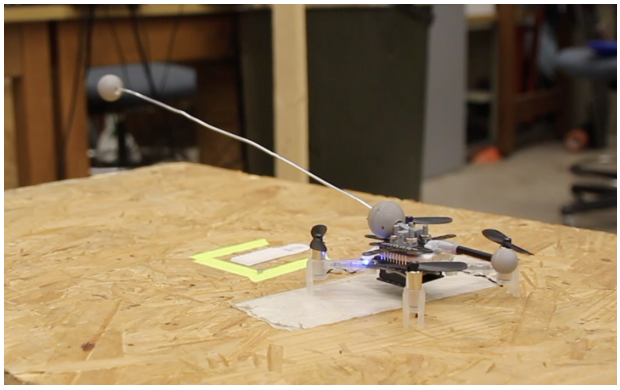
The Crazyflie 2.1 is a lightweight quadcopter drone. It has a basic mass of 27 g, and it has a measured mass of 32.4 g with battery installed. The drone uses C++ firmware code and Python client libraries, which are available publicly on GitHub as open source repositories. The drone has a maximum recommended payload mass of 15 g, which is taken into consideration in the design of the pendulum and motion capture hardware. The drone links to a long-range antenna for reliable communication and control. This antenna uses a standard USB interface, and the antenna and client software can be used across a wide range of platforms [6].

B. Pendulum

The pendulum is implemented in hardware using an aluminum pendulum and a hinge mechanism. The hinge constrains the pendulum to one degree of freedom, which simplifies the system dynamics and is consistent with the theoretical model. The hinge components are 3D printed from polylactic acid (PLA) filament and attached to the Crazyflie drone by screwing it onto the motion capture marker deck. The pendulum itself is constructed from a thin aluminum wire and attached to the hinge using super glue. The first iteration of the pendulum measured 30 cm tall. Later in testing, a longer pendulum measuring XX cm was used to improve controllability. The drone-pendulum system can be seen below in Figure 3.



(a) Hinge Mechanism

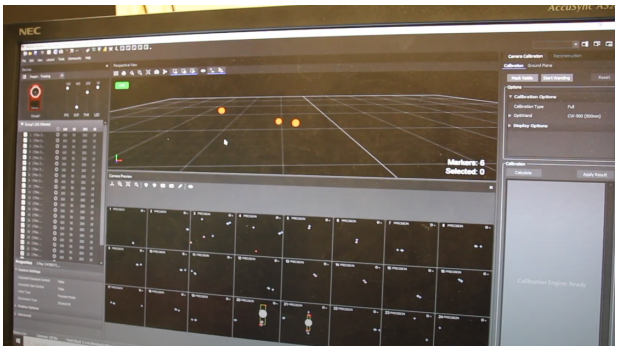


(b) Drone with Pendulum

Fig. 3 Drone-Pendulum System

C. Motion Capture System

In order to observe the angle of the pendulum, a motion capture system was used. The motion capture system used was developed by the OptiTrack company. This system contains an array of OptiTrack Flex 3 cameras and the OptiTrack Motive motion capture software. Images of the system can be seen below in figure 4.



(a) Motive Motion Capture Software

(b) Flex 3 Camera

Fig. 4 OptiTrack System

The goal of using the OptiTrack Motion Capture system was to retrieve the angle of the pendulum with respect to the ground in the world frame. This meant the need for at least two reflective motion capture balls, one at the base of the pendulum and one at the tip. The angle of the line formed by the two balls could be found by using the x, y, and z positions of each ball, as seen in equation 22. The angle retrieved would be 0 degrees when the pendulum is fully vertical and +/- 90 degrees when full horizontal.

$$\alpha = \arctan\left(\frac{z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}\right) \tag{22}$$

Although the magnitude of the angle with respect to the ground was correct when tested, the sign on the angle was not correct. In the drone-pendulum system, a positive pendulum angle is considered when the pendulum falls in the positive x-direction of the drone. In order to retrieve the sign of the angle, a third reflective ball was attached to the drone in the drone's positive x-direction, serving as a reference point. Through the use of the reflective balls, two vectors can be formed. Vector 1 is considered to be from the base of the pendulum to the tip of the pendulum. Vector 2 is considered to be from the base of the pendulum to the reference point. By calculating the component of Vector 1 onto Vector 2, the sign of the pendulum can be found. If the calculated component is positive, the pendulum is falling in the drone's positive x-direction with a positive pendulum angle. If the calculated component is negative, the pendulum is falling in the drone's negative x-direction with a negative pendulum angle. Since the reference ball moves as the drone pitches, the z position of the reference ball is corrected so the reference vector is parallel with the floor. This ensures the direction calculation will be correct no matter the pitch of the drone.

The OptiTrack system itself is contained within a network of cameras and a central computer. The x, y, and z positions of each marker is then sent from the central computer to a drone operator's laptop using Python and network protocols. The data is then packaged and sent from the drone operator's laptop to the drone using radio communication. The drone is then able to unpack and process this data to use in the controller.

In perfect conditions, the observation of the pendulum angle worked well. However, with the nature of the drone-pendulum motion capture setup, any third parties using reflective balls or tape would interfere with the motion capture system.

VIII. Results

In this section, results from both simulation and real-world testing are reported. The controller is shown to be successful at controlling the drone-pendulum system in simulation.

A. Simulation Results

The simulation produces effective stabilization of the pendulum. With an initial condition of the pendulum of -0.05 rad from vertical, the drone is able to right the pendulum, as shown in figure 5. An interesting but expected result of x-position error can be seen in the uppermost plot. In trying to steer the pendulum back to vertical, the entire drone-pendulum system must move in x-direction. This indicates an interesting behavior of the system. From the initial negative pendulum angle, the drone travels in the negative x-direction, which causes the pendulum to flip to a positive angle, and the drone then travels in the positive x-direction as a correction. The steady increase in x-position error over time is a possible indication that more tuning of the controller is necessary, such as increasing the weight on the x-position error.

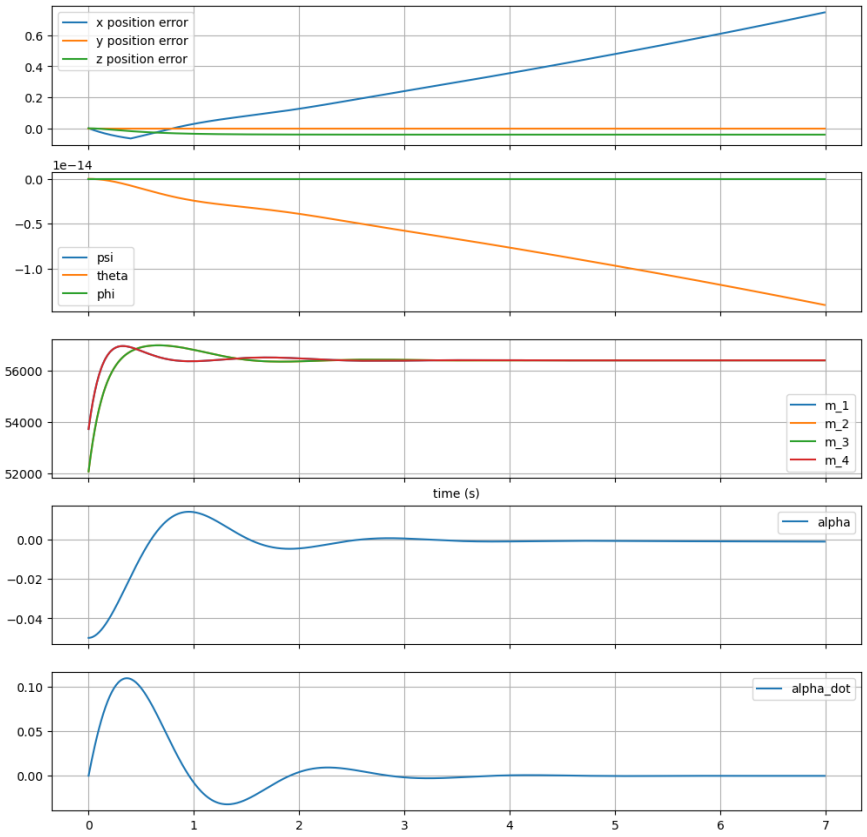


Fig. 5 Simulation Test

Using the simulation demonstrates that the dynamical system derivation and LQR method can produce a working controller for the system. Furthermore, this method of designing a controller is shown to reliably create working controllers with a variety of weights, and the controller is shown to reliably work with non-zero initial pendulum angles. It is seen in the middle plot in figure 5 that the motor commands converge quickly and do not saturate, indicating that the drone's performance is sufficient to carry the pendulum.

B. Data Analysis

Using the initial controller developed in simulation, flight tests were performed to verify the controller's ability to balance the pendulum on the drone. Through these flight tests, the controller was tuned through the LQR process to push towards the desired behavior. Due to the nature of the system, many of the flight tests performed resulted in unusable data. However, there are several flight tests that showed the expected behavior even without balancing the pendulum. In the section below, the analysis of four flight tests will be performed.

Parsing and visualizing the data collected during the flight test, it was determined that the most important variables for understanding the drone behavior were the angle of the pendulum, the x-position of the drone, the z-position of the drone, and the pitching angle of the drone. Unfortunately, the pitching angle of the drone was saturated for most test flights as the drone would need to make large corrections for the pendulum angle. Visualized in the plots below are the variables mentioned above except pitching angle of the drone.

Taking a look at figure 6, the results of Flight Test 1 can be seen. The flat portions of the data represent the time where the drone was in position for flight. For this particular flight, the drone started at a z-position of 1.175 meters off the ground. The end of the test is when the drone impacts with ground, shown when the z-position reaches 0 meters. Immediately from the drop of the drone, the pendulum angle increases as the pendulum falls. The drone reacts by trying to catch the pendulum, seen by the left to right movement in the x-position. The drone was not able to correct fast enough and eventually the pendulum hits the stopper on the drone, seen by the large drop in angle around 1 second into the flight. This causes the drone to lose all control and crash into the ground.

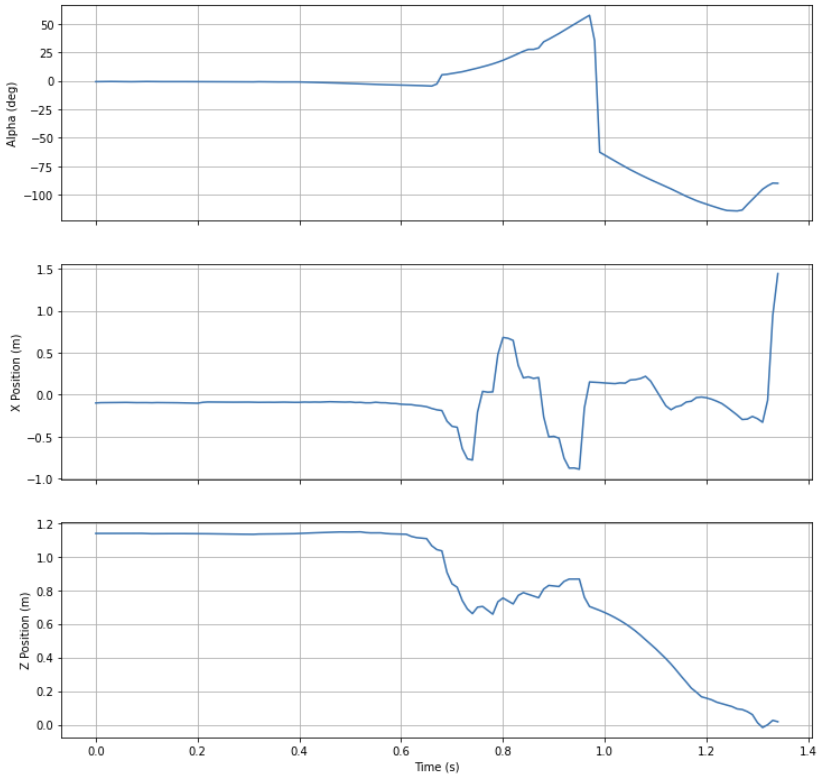


Fig. 6 Flight Test 1

Similar to Flight Test 1 and looking at figure 7, Flight Test 2 shows similar behavior with similar controllers. Flight Test 2 used a controller where the angle of the pendulum had a high priority, however looking at the data, it does not seem to affect the end results too much.

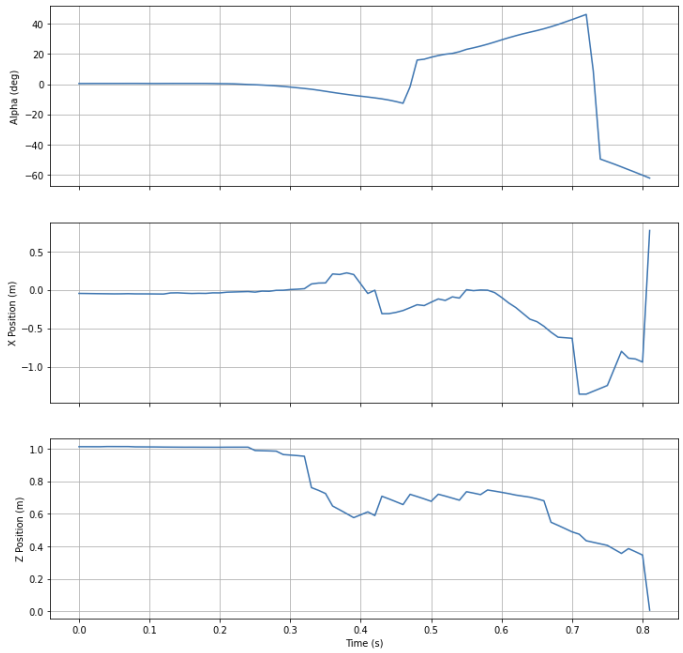


Fig. 7 Flight Test 2

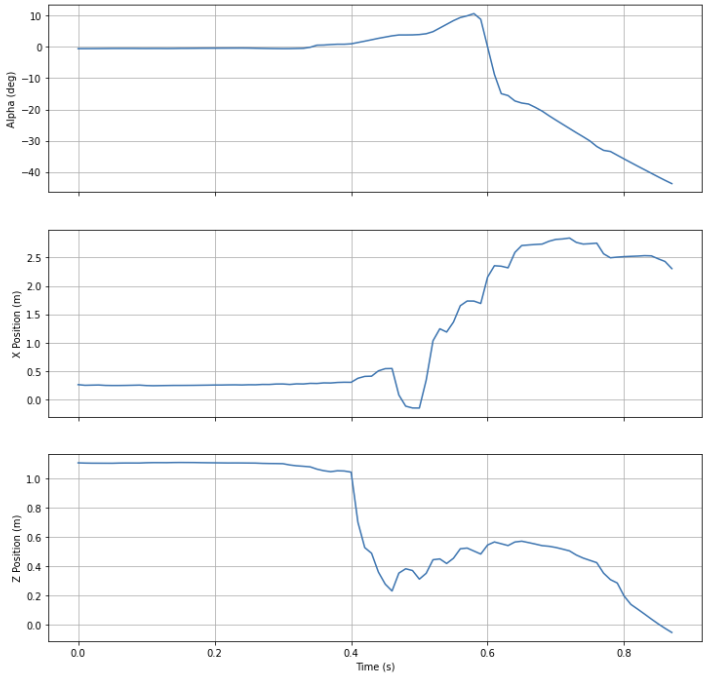


Fig. 8 Flight Test 3

Similar to Flight Test 1 and looking at figures 7 and 8, the behavior of the drone is repeatable and the drone seems to be making the correct decisions for how to balance the pendulum on the drone, but is not able to correct its trajectory fast enough before failure points. There are two failure points that were noticed during the many flight tests ran with the drone-pendulum system. First, as the drone is trying to correct its trajectory, there is not enough lift to maintain a stable height, causing the drone to eventually crash into the ground. Second, the pendulum falls and hits

the stopper on the drone, creating a large moment on the drone and causing it the crash. In conclusion, by looking at the data, the drone is trying to catch and balance the pendulum but is not able due to physical constraints with the current version of controller. With more experimentation, a controller could be developed to balance the pendulum for this specific drone.

C. Trajectory

The addition of the pendulum results in a modification to the standard response of the drone to a given equilibrium state. Specifically, as can be seen in the flight data, the drone must first move in the opposite direction to the desired position, and then proceed to move in the correct direction to both reach its desired position and correct the pendulum back to an upright position. Just directly traversing towards the desired position would result in the pendulum no longer being in its desired position. This is a counter-intuitive behavior that the team found fascinating, as it was different from the standard behavior of the drone seen in class. The drawback of this was that this controller wouldn't allow for the drone to take off, and so the pendulum had to be started at its equilibrium height.
This behavior is characteristic of pendulum-on-cart systems. However, because the drone had to pitch to move in the x direction, it also began losing altitude. This was observed during tests, as highlighted in Figure 9, and also in the data shown above.

Fig. 9 Trajectory of Drone while dropping

During these tests, the team noticed an interesting parallel with SpaceX's Starship's landing maneuver. Since Starship also involves creating large pitch changes to an "inverted pendulum" to bring it back to an upright position. Figure 10 from the Starship SN8 test flight on December 2020 shows how Starship also had to move in the opposite direction to flight itself to an upright position before landing

Fig. 10 Starship SN8 Trajectory

IX. Conclusion

Implementing a flying version of the classical inverted pendulum problem provided an excellent opportunity to explore a complex dynamical system. The Motive Capture system proved a viable method for observing the pendulum, allowing the drone to reliably receive pose data. While the system displayed expected behaviors, the pendulum was never able to fully balance. The added weight of the pendulum on the drone decreased it's agility and it was not able to match the speed and scale of movements necessary to control the pendulum, despite its capability of quick translational accelerations. The aluminum pendulum was also not stiff enough and was constantly bent out of shape, deviating from the expected design.

By looking at the behavior of the drone on a small scale, the drone's flight path matched the characteristic behavior of other inverted pendulum systems, such as a pendulum-on-cart system. Even though the drone was not fast enough to correct the trajectory before failure, the drone is accurately attempting to balance the pendulum and making effective adjustments.

An improved implementation of the drone-pendulum system will be pursued after the conclusion of this project. The major modifications will be the use of a larger, more powerful drone and a stiffer pendulum design. Additionally, an inertial measurement unit (IMU) will be used in place of the Motive Capture system and more time will be devoted to finely tuning the controller. This will allow the system to be used outside of the drone laboratory and provide pose data with a smaller delay. The overall goal is to use the working system as an educational outreach tool to expose future engineers and students to the idea of aerospace control systems and provide them with accessible opportunities to explore those interests. The system will be shown at the University of Illinois at Urbana-Champaign's Engineering Open House , where it will be used to explain how rockets and aerial robots function and control themselves to K-12 students from across Illinois.

Acknowledgments

The authors would like to thank Tim Bretl, Alen Golpashin, Jongwon Lee, and all other course staff for their guidance. The team would also specially like to thank Professor Bretl for his continued help with deriving the equations

of motion of the drone-pendulum system. They would also like to thank the laboratory manager, Dan Block, for guidance with the MoCap system, and Scott Bout for his guide on implementing the MoCap system with the Crazyflie.

References

[1] Hehn, M., and D'Andrea, R., "A flying inverted pendulum," *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 763–770. https://doi.org/10.1109/ICRA.2011.5980244.

[2] Benson, T., "Model rocket stability," , 2021. URL https://www.grc.nasa.gov/WWW/k-12/VirtualAero/BottleRocket/airplane/rktstab.html.

[3] Kafetzis, I., and Moysis, L., "Inverted Pendulum: A system with innumerable applications," 2017.

[4] "Motion capture for Robotics," , 2022. URL https://optitrack.com/applications/robotics/.

[5] Ahmed, M., English, L., Mendez, E., and Moore, A., "Long Exposure Light Tracing of a Fully-Observable Drone System," , Fall 2021. AE483 Final Project Report, University of Illinois at Urbana-Champaign.

[6] "BitCraze Crazyflie 2.1," , 2022. URL https://www.bitcraze.io/products/crazyflie-2-1/.

Appendix A: Group Member Contributions

Day	Task	Contributor(s)
11/02/2022	Wrote topic paper	Anshuk
11/02/2022	Designed 1st iteration of pendulum hinge design in NX	Rishi
11/02/2022	Derived new equations of motion for combined system	Anshuk
11/02/2022	Purchased pendulum components from store	Harry, Anshuk
11/03/2022	Wrote Abstract, Introduction, and Milestones for report	Bella
11/03/2022	Implemented 1st iteration of pendulum hinge design	Rishi, Anshuk
11/03/2022	Designed 2nd iteration of pendulum hinge design in NX	Rishi
11/03/2022	Implemented 2nd iteration of pendulum hinge design	Rishi
11/03/2022	Assembled the pendulum hinge design	Rishi, Bella
11/03/2022	Met with Dan Block to understand the MoCap system in Transportation 302	Rishi, Anshuk, Harry
11/03/2022	Modified pendulum to fit MoCap markers	Rishi, Anshuk, Harry
11/04/2022	Proofread plan	Bella, Rishi, Anshuk, Harry
11/04/2022	Submitted plan	Bella
11/09/2022	Created PyBullet simulation and modeled drone with pendulum URDF	Harry
11/18/2022	Debugged angle estimation for MoCap script	Rishi, Anshuk, Harry
11/22/2022	Designed first iterations of controller	Anshuk
12/01/2022	Setup Drone and MoCap system for flight testing	Rishi, Bella
12/02/2022	Flight tested drone-pendulum system	Rishi, Anshuk, Bella
12/03/2022	Flight tested drone-pendulum system	Rishi, Anshuk
12/04/2022	Re-derived corrected equations of motion	Anshuk
12/04/2022	Measured new drone-pendulum system mass and calculated new moments of inertia	Harry
12/05/2022	Flight tested drone-pendulum system	Rishi, Anshuk, Harry
12/07/2022	Filmed and edited video	Bella, Rishi
12/09/2022	Flight tested drone-pendulum system	Rishi, Anshuk
12/10/2022	Flight tested drone-pendulum system	Rishi, Anshuk
12/11/2022	Flight tested drone-pendulum system	Rishi, Anshuk
12/14/2022	Wrote sections III and IV, cleaned up code for submission	Anshuk
12/14/2022	Wrote section V and VII.C	Rishi
12/14/2022	Wrote section VI and VII.A and cleaned up code for submission	Harry
12/14/2022	Wrote section VII.B and the conclusion	Bella
12/14/2022	Proofread report	Bella, Rishi, Anshuk, Harry

Appendix B: Code Repository

The project GitHub repository can be found at the following link: https://github.com/rpate392/DroneInvertedPendulum.