A Novel Control System for Autonomous Quadcopter Navigation

Statement of Problem

Quadcopter drones are growing in popularity due to their versatility in delivery, organ transportation, and serving as communication base stations. However, most drones are remote controlled because autonomous control or “self-driving” quadcopters are very difficult to stabilize and navigate. This project seeks to improve autonomous quadcopter stability and navigation by 1) developing a novel dual loop control feedback system, 2) simulating and modifying control algorithms to work in the dual loop control feedback system using original MATLAB programs and 3) building two physical prototypes (Single Axis and Quad Axis) to verify the results from the simulation and to learn how autonomous quadcopters work.

Introduction

- Unmanned aerial vehicles (UAVs) are popular in both commercial and military applications (delivery, search-and-rescue, communication), but autonomous control of UAVs are difficult to stabilize and navigate, which limits their widespread usage.
- Most UAVs are manually controlled because autonomous drones have complicated aerodynamics, due to six states of motion (positional x, y, z, and rotational ϕ, θ, ψ) controlled by only four motor inputs, making the system over-actuated. This makes UAV stability while hovering and in motion a challenge.
- To stabilize UAVs, various control system algorithms are in early-stage development. The four algorithms used for a feedback control system are the Proportional-Derivative (PD), Proportional-Integral-Derivative (PID), Back-stepping Control (BSC), and Sliding Method Control (SMC).
- This project innovates a novel dual loop feedback control system and uses a sequence of optimization steps to stabilize and navigate autonomous quadcopters, a type of UAV.
- Quadcopter dynamics requires an understanding of the of how the angular velocity of the motors creates thrust and torque that cause the quadcopter to move with 6 degrees of freedom in both translation {x,y,z} and rotation {ϕ,θ,ψ} as seen in Figure 1.

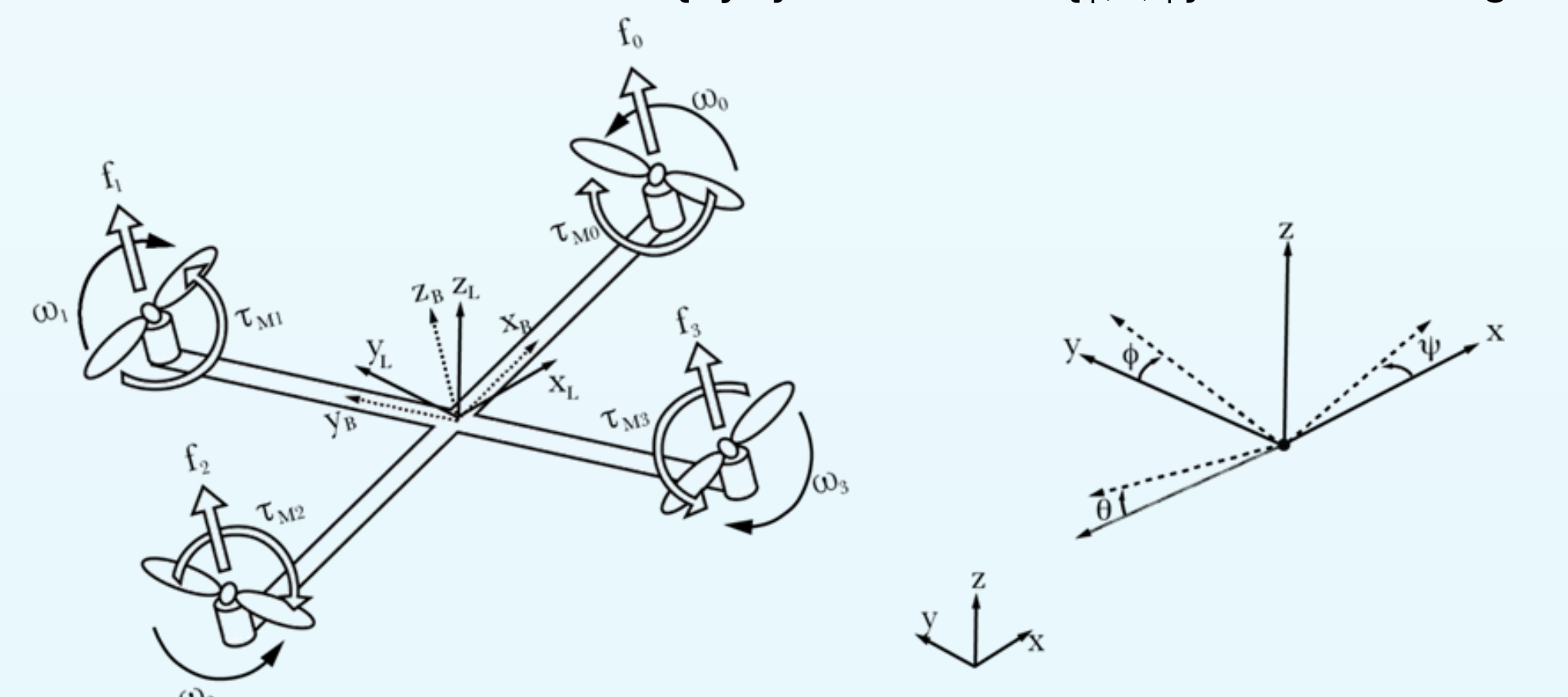


Figure 1. (a) Quadcopter Dynamics and its Reference Frames

- The Thrust, T_B, is the force to keep the quadcopter in the air against gravity. It is a function of motor angular velocity, ω_i, and observed lift constant, k. (Eq 1a) The torques, τ_B, rotate the quadcopter so the quadcopter can move in the right direction. Three torques in each angle are a function of motor angular velocity, ω_i, the quadcopter dimension, l, empirical drag and lift constants, b and k. (Eq 1b)

$$T_B = \begin{bmatrix} 0 \\ 0 \\ T = \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \qquad \tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^{4} \tau_{Mi} = b \sum_{i=1}^{4} (-1)^i \omega_i^2 \end{bmatrix} \tag{1}$$

(a) (b)

- Using Newtonian translational and rotational physics with the mass, m, and Inertia, I, of the quadcopter, the thrust and torques can provide the x,y,z acceleration and ϕ,θ,ψ angular acceleration of the quadcopter as shown in Eq 2.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi sin\theta cos\phi - cos\psi sin\phi \\ cos\theta cos\phi \end{bmatrix} \qquad \begin{bmatrix} \alpha_{B\phi} \\ \alpha_{B\theta} \\ \alpha_{B\psi} \end{bmatrix} = \begin{bmatrix} \tau_\phi / I_{xx} \\ \tau_\theta / I_{yy} \\ \tau_\psi / I_{zz} \end{bmatrix} \tag{2}$$

(a) (b)

- The dynamics of the quadcopter are simulated via original programming in MATLAB (Fig 1b) using the equations above to show the relation of the angular movements in the quadcopter to move it in the x and y directions. When there is no feedback loop, any amount of noise or disturbance will make the quadcopter become unstable and crashes, which makes a quadcopter feedback control system important. A control system is shown in Fig 2.

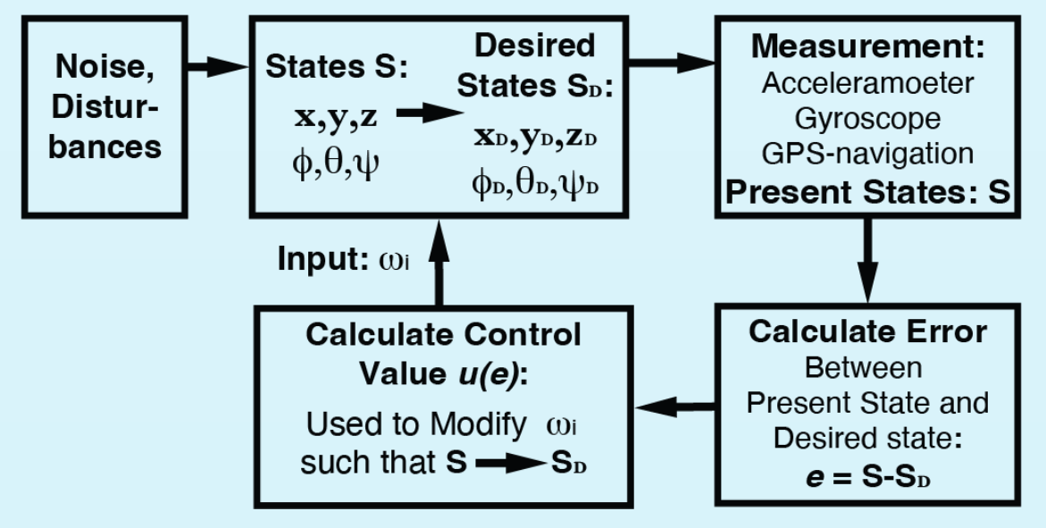


Figure 2. Feedback Control System for Autonomous Quadcopter

- A control system is required to change a state of a system, S, to a desired state, S_D, with sensor information. Error, e, from the desired state and the present state is calculated and modified to provide an input to the system. For the quadcopter, the input is needed to control the angular velocities of the motors. This process is represented in Fig 2.
- A single loop controller (Fig 3) can be function, but parameters for navigation must be very small to minimize change at every step because the quadcopter is an under-actuated system. This limits stabilization and navigation, which degrades performance.

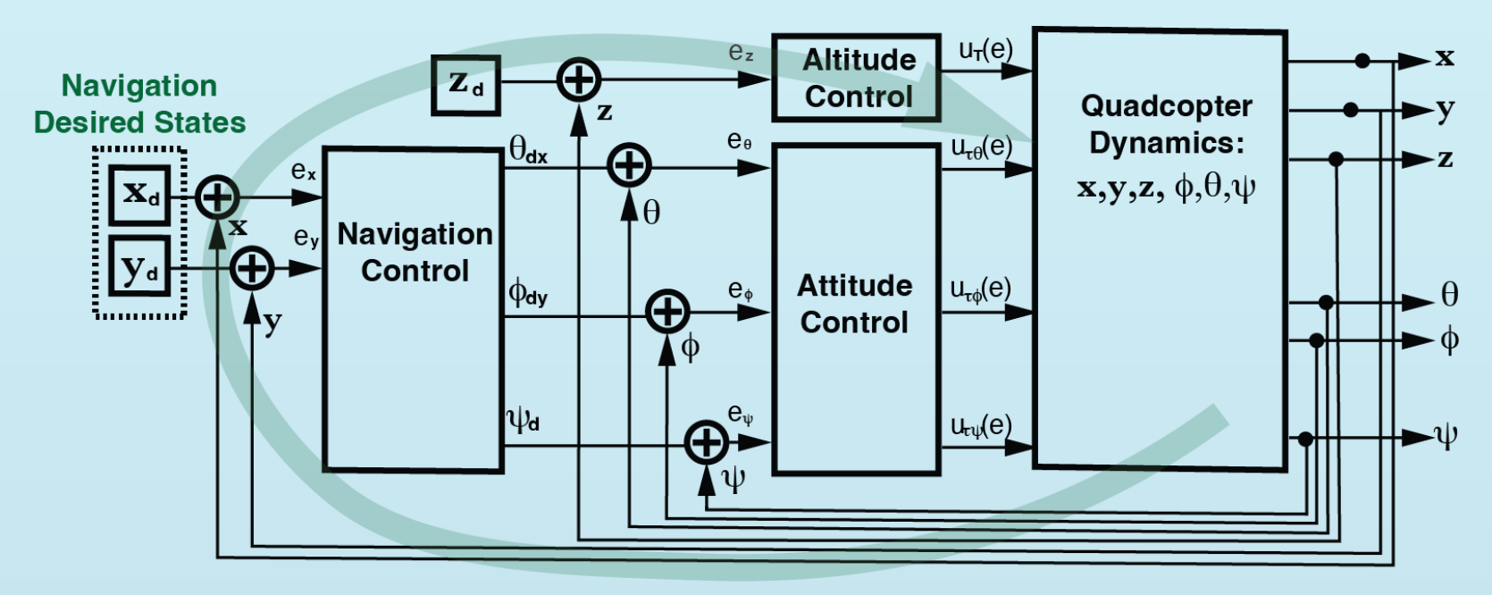


Figure 3. Single Loop Control System for Autonomous Quadcopter

Methodology

- Quadcopter dynamics were simulated using original programming in MATLAB
- A Novel Dual Loop Control Feedback System is developed for Quadcopter Stabilization and Navigation (Fig 4)
  - The Dual Loop Control System contains 2 independent feedback loops to control hover dynamics and navigation dynamics independently
- Original MATLAB programs were written to simulate four control system algorithms modified to be used in the Dual Loop Control System: the Proportional-Derivative (PD), Proportional-Integral-Derivative (PID), Back-stepping Control (BSC), and Sliding Method Control (SMC).
  - For each modified control algorithm simulation, transient and noise data was collected for each parameter: motor angular velocities, translation motion, angular motion, and angular velocities of the quadcopter.
  - Transient and noise criteria data was analyzed to calculate cost functions based on overshoot of position, settling time, dynamic range of angles and angular velocities, and standard deviation of the noise and disturbance for each state
- Using the cost functions, Control Parameters for each of six states were selected to optimize the 4 Modified Control Algorithms (24+ cost functions total). An additional Cost function was used to compare the performance of 4 optimized algorithms.
- A Single Axis Prototype (Fig 5) was built to understand how a Quadcopter functions and how to implement the simulated control theory in real world conditions.
  - Original programs in Python were written to control the Gyroscope and Brushless motors
  - An oscilloscope (Fig 6) was used to validate the PWM signals that were required to set the correct motor gains
- A Quad Axis Prototype (Fig 7) was built to further study control theory with more states (z, ϕ, θ, ψ) in real world conditions
  - Original programs in Micro Python were written to control the Quad axis prototype with a microcontroller.
  - HTML was used to create a webpage for autonomous control of the prototype.
- Angular state data was collected from prototypes to analyze difference between simulated models and real-world data.

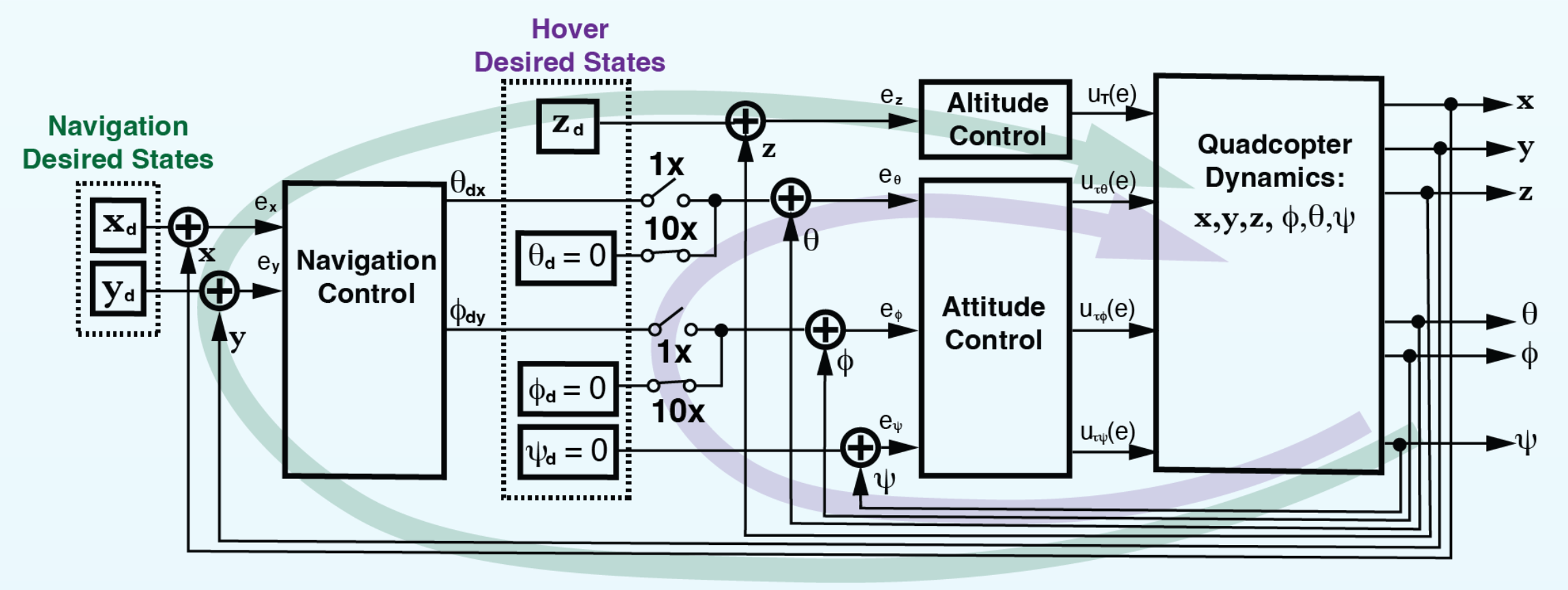
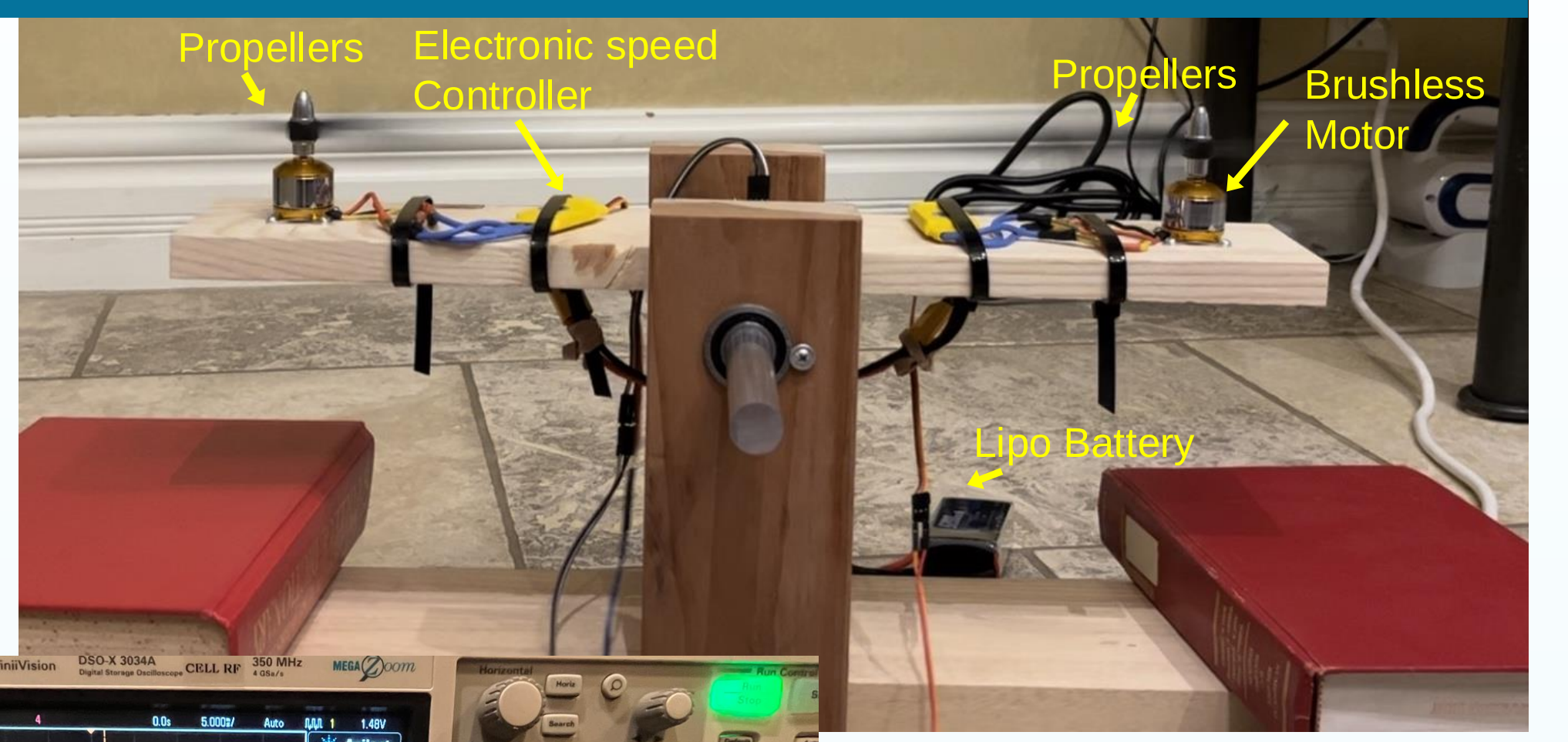


Figure 4. Novel Dual Loop Control System for Quadcopter



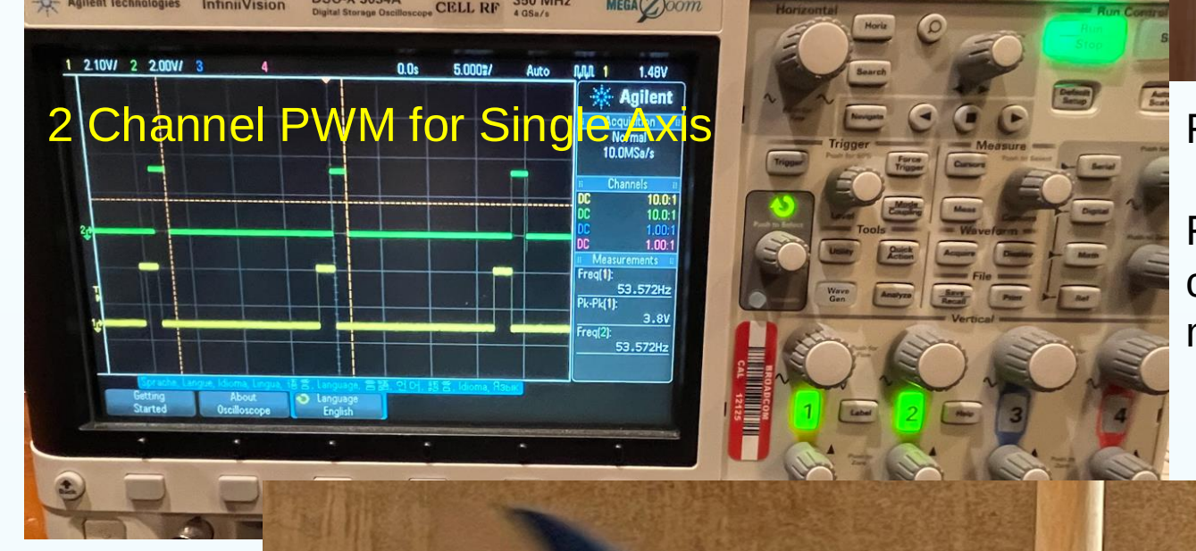


Figure 5. Single Axis Prototype (up)

Figure 6. Oscilloscope with PWM control signals required for the DC motors (left)

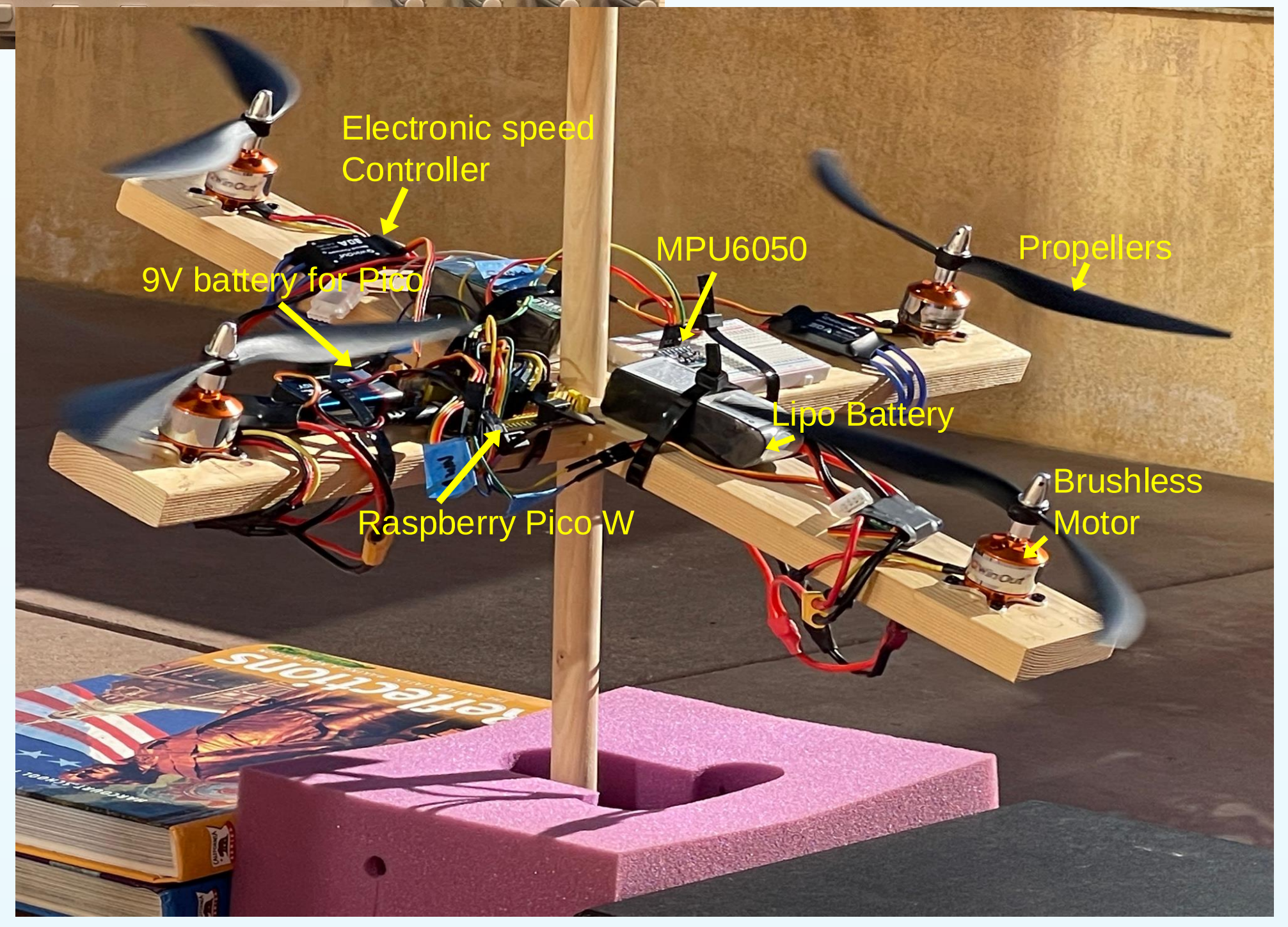


Figure 7. Quad Axis Prototype

Simulation Data for Quadcopter Navigation Dynamics

Proportional-Derivative (PD) Control Algorithm

A PD controller relies on the error of the present state, $e(t)$, the proportional term, and the derivative of this error to create a control input, $u(t)$ to the quadcopter to get the quadcopter to the desired state.

$$e(t) = x_d(t) - x(t)$$
$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt}$$

	Nav	Nav	Hover	Hover	Hover
	x,y	ϕ,θ	ϕ,θ	ψ	z
Kp	8	8	6	8	4
Kd	6	8	2	4	4

PROs:
- Simple Control System to Implement
- No integration is required only proportional and derivative in algorithm
- Fast settling for angles

CONs:
- Cannot handle Steady State errors such as imbalance in motors gain or weight distribution
- Cannot tolerate high levels of noise or disturbances

Data for 20m step in x- and y- directions

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	12s	3.5s	3.5s	3s	4s
Noise (St Dev)	1.5m	0.1m	4.4°	2.3°/s	4.4°
Maximum Range			35°	80°/s	15°

Proportional-Integral-Derivative (PID) Control Algorithm

A PID controller is like a PD control but includes an integral term that can remove static errors like motor gain difference. Most common controller.

$$e(t) = x_d(t) - x(t)$$
$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

	Nav	Nav	Hover	Hover	Hover
	x,y	ϕ,θ	ϕ,θ	ψ	z
Kp	4	8	8	8	5
Ki	1	4	4	4	1
Kd	4	4	4	6	4

PROs:
- Simple Control System to implement but requires integration
- Can handle Steady State errors such as imbalance in motors gain or weight distribution

CONs:
- Cannot tolerate high levels of noise or disturbances
- Slow settling for angles and angular velocity

Data for 20m step in x- and y- directions

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	11.5s	3.5s	12.5s	10s	10.5s
Noise (St Dev)	1.5m	0.1m	4.3°	2.1°/s	5.1°
Maximum Range			35°	80°/s	15°

Backstepping Control (BSC) Algorithm

A BSC controller assumes a local stable point by changing the error term to be stable. Error term includes an error for the state and its derivative. Back-stepping over corrects and then removes overcorrection in the next step.

$$s(t) = (\dot{x}_d(t) - \dot{x}(t)) + K_1(x_d(t) - x(t))$$
$$u(t) = K_1(s(t) + K_I e(t) - K_2 s(t-1))$$

	Nav	Nav	Hover	Hover	Hover
	x,y	ϕ,θ	ϕ,θ	ψ	z
K1	0.5	16	8	8	4
K2	1.5	12	4	4	4

PROs:
- Very fast settling for angles to get back to hover state
- Can tolerate high levels of noise or disturbances

CONs:
- More complex algorithm to implement
- Angular Velocity and Motor Speeds requirements are too high for changes in x,y direction in the real-world.

Data for 20m step in x- and y- directions

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	8.5s	3s	1.3s	0.9s	1.8s
Noise (St Dev)	0.2m	0.1m	3.9°	5°/s	4.0°
Maximum Range			65°	850°/s	18°

Sliding Method Control (SMC) Algorithm

Similar to BSC, the SMC controller assumes a local stable point by changing the error term to be always stable. The sliding method uses a signum (or sign function) to always make the correction in the opposite direction of the error.

$$s(t) = \dot{x}_d(t) - \dot{x}(t) + K(x_d(t) - x(t))$$
$$u(t) = K^2 e(t) + K_1 \operatorname{sgn}(s(t)) + K_2 s(t)$$

	Nav	Nav	Hover	Hover	Hover
	x,y	ϕ,θ	ϕ,θ	ψ	z
K	0.5	2	2	1	2
K1	3	20	4	1	1
K2	3	8	4	3	5

PROs:
- Very fast settling for angles to get back to hover state
- Tolerates high levels of noise or disturbances
- Maximum angle change and angular velocity is low during change in position
- Tolerates steady state error (imbalance) well

CONs:
- Most complex algorithm to implement

Data for 20m step in x- and y- directions

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	8s	3s	1.7s	1.2s	2.3s
Noise (St Dev)	0.3m	0.1m	3.9°	7.5°/s	4.1°
Maximum Range			30°	80°/s	10°

Figure 8. Single Loop Navigation Dynamics with imbalance, noise and disturbance

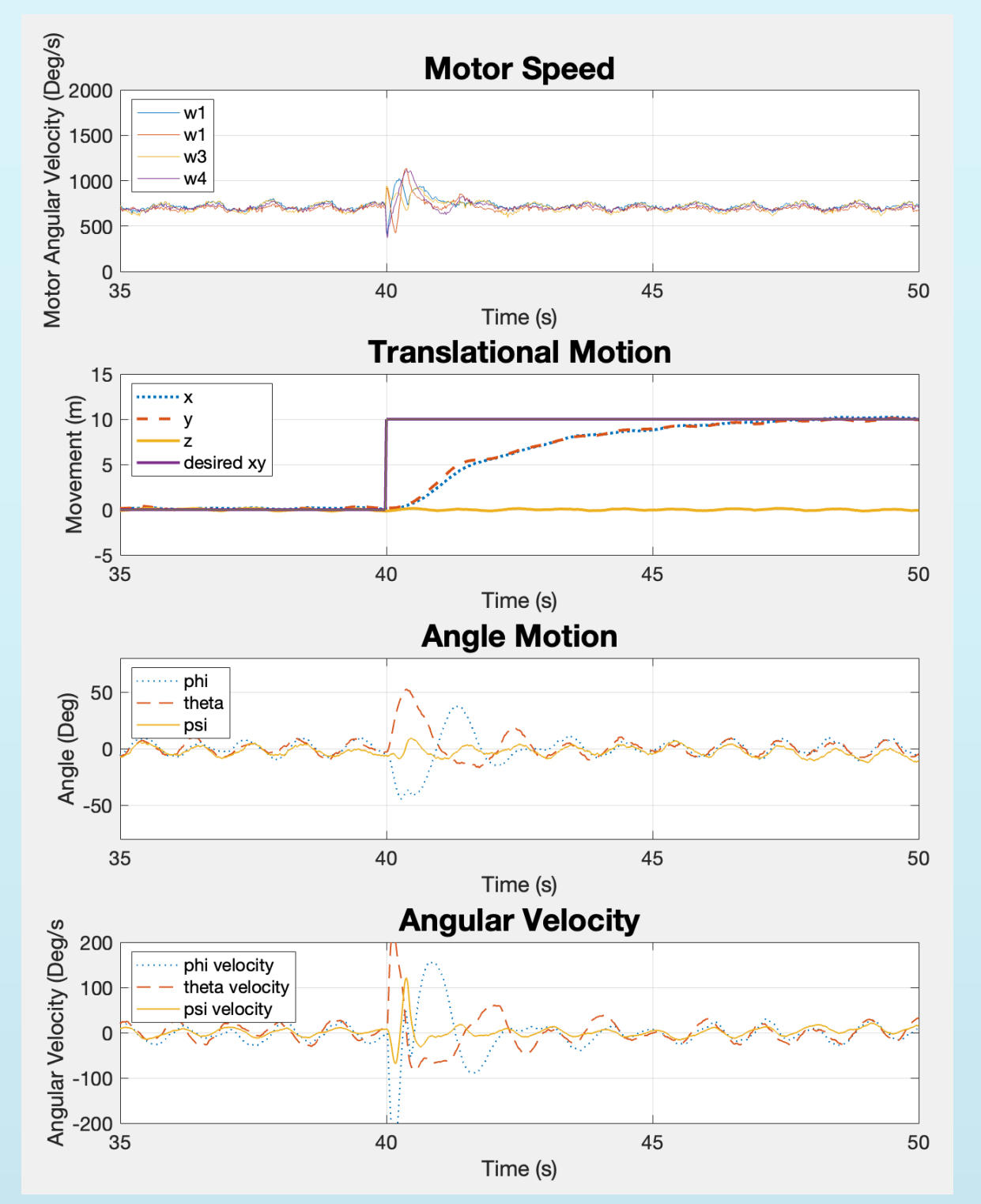


Figure 9. Dual Loop Navigation Dynamics with imbalance, noise and disturbance

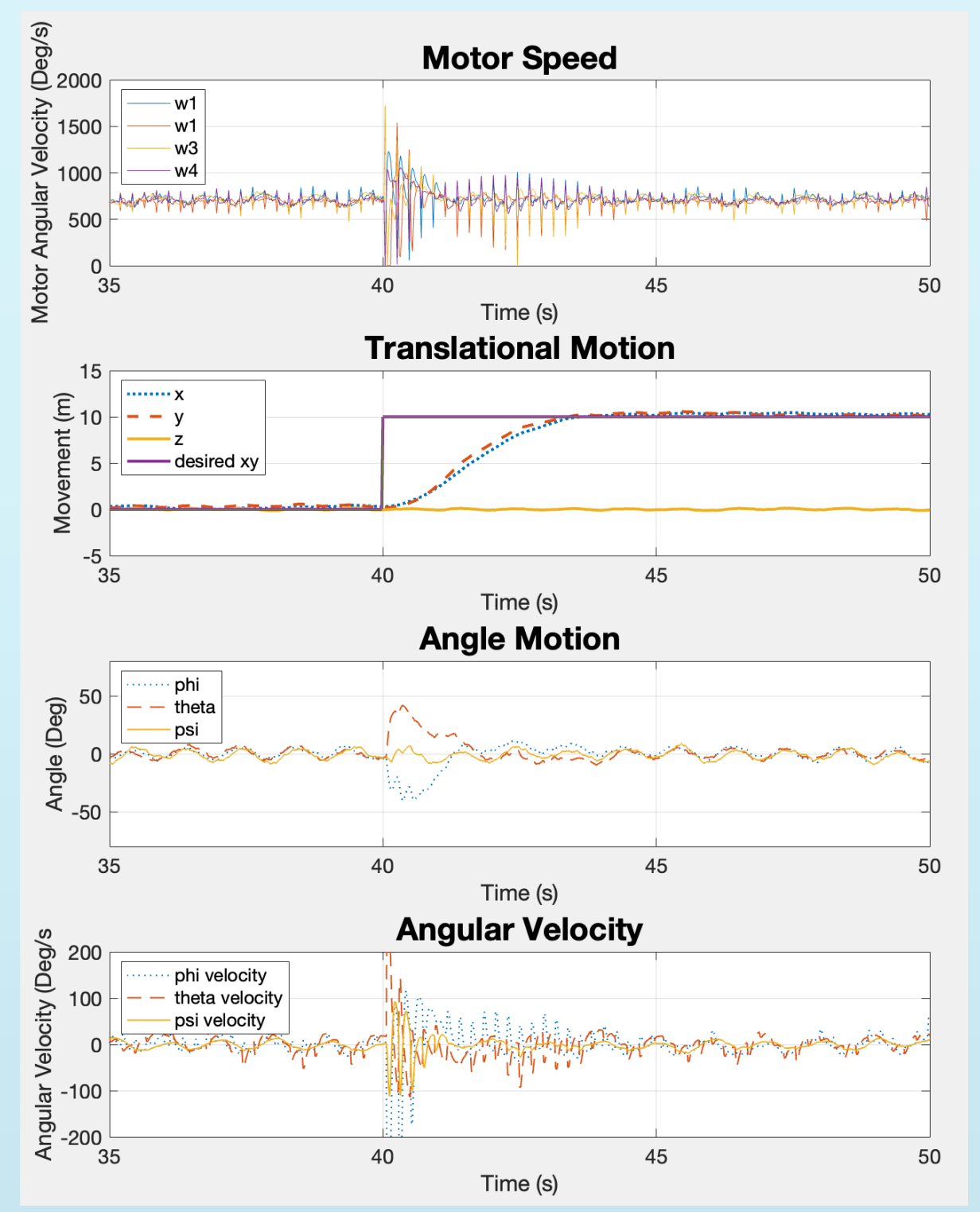


Figure 10. Single Loop Navigation Dynamics with imbalance, noise and disturbance

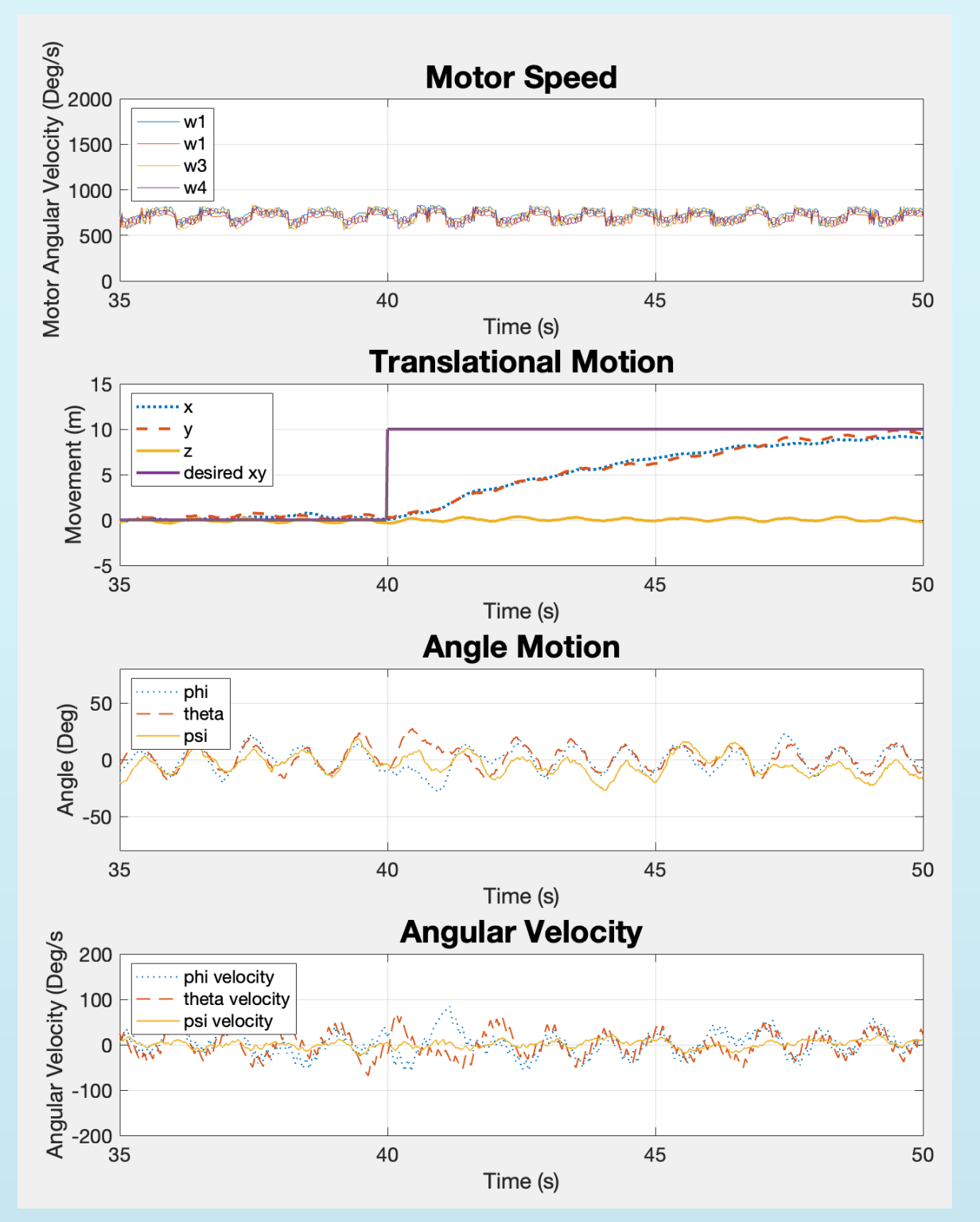
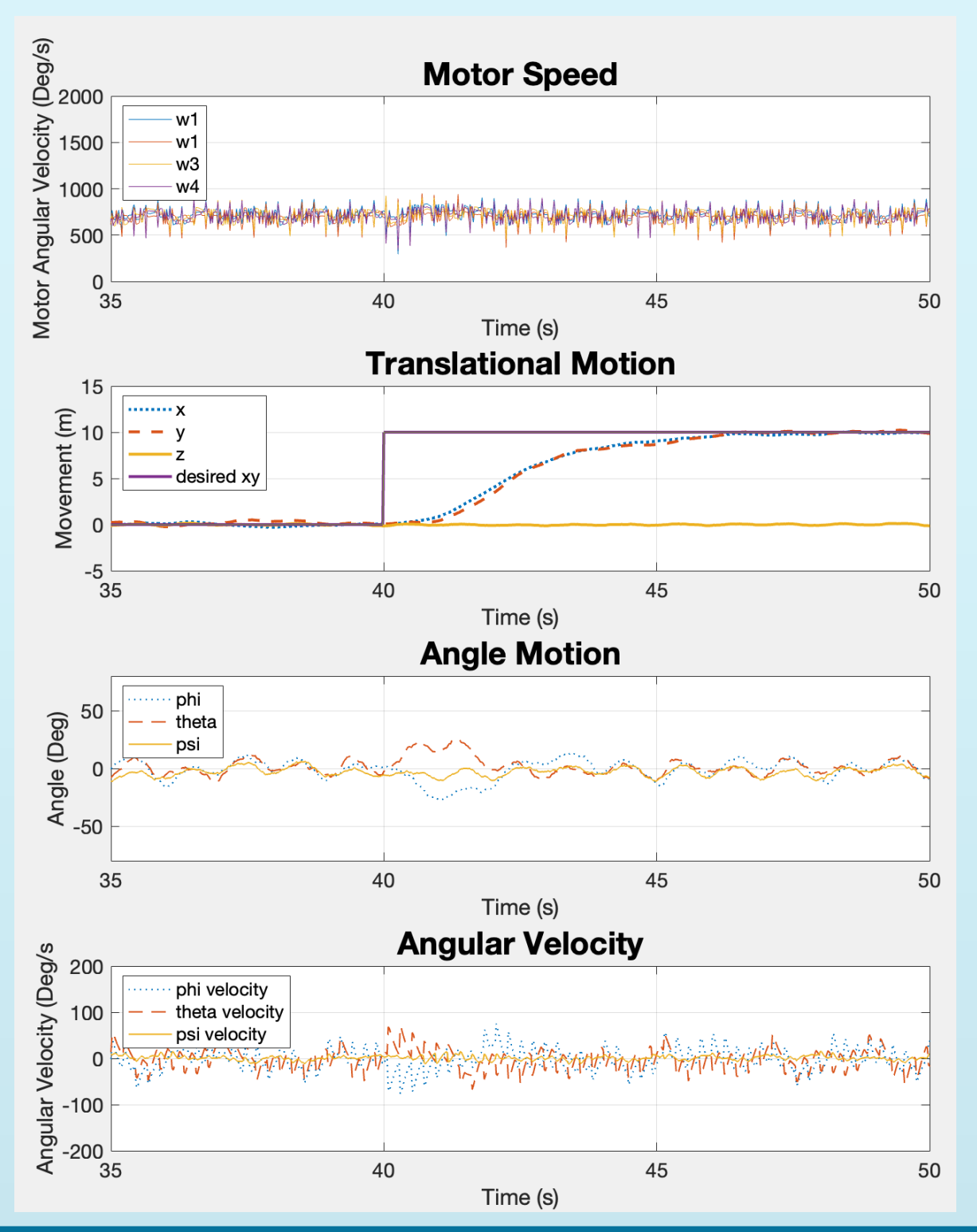


Figure 11. Dual Loop Navigation Dynamics with imbalance, noise and disturbance



Results from the Prototypes

Single Axis Prototype

- Used to learn and explore the inner workings of a quadcopter and feedback control system: motors, propellors, ESCs, gyroscope, and its communication with the Raspberry Pi microprocessor using I^2C and PWM signals.
- Tests a simplified control algorithm for a single state variable {θ} with various parameter values for the Control input.
- Results from prototype validate the simulated control models: parameters and angular velocities from optimized simulation and prototype are the same. Optimized parameters {Kp=6,Kd=2} are better than random {Kp=3,Kd=0.5} in terms of angle stabilization over time (Fig 12). [single axis data collected from CSV data logging]

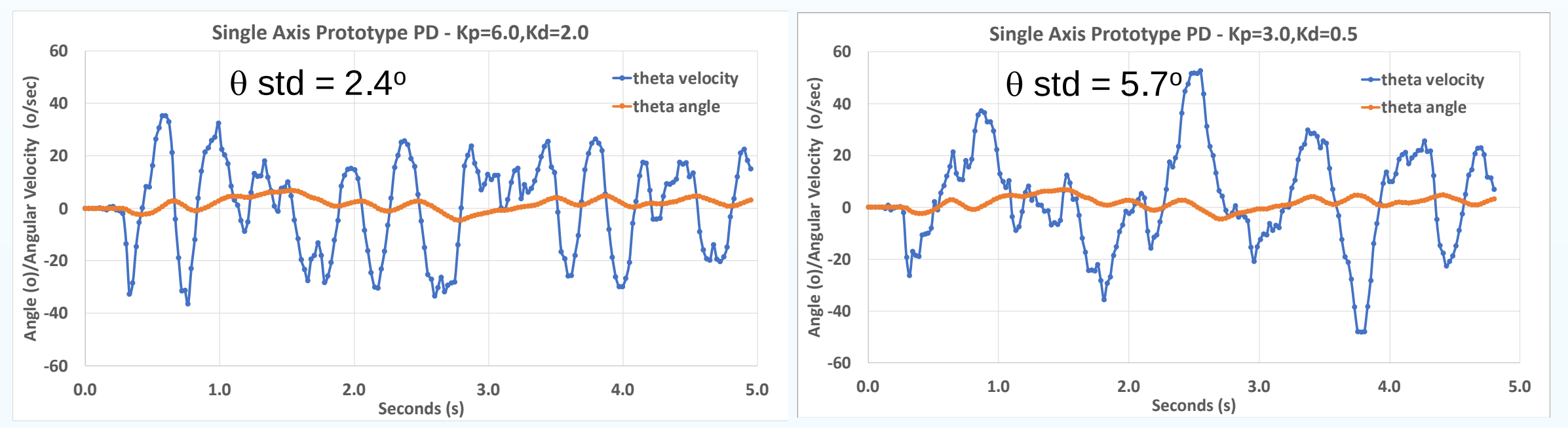


Figure 12. Single Axis Angle and Angular Velocity Data during Flight

Quad Axis Prototype

- More complex system compared to the single axis prototype allows for 4 degrees of motion {ϕ, θ, ψ, z} which defines the altitude and attitude of the quadcopter.
- Consists of 4 wooden arms and a center hole to keep the Quad Axis restricted in the x- and y- direction. It allows rotational freedom {ϕ, θ, ψ} and hover in the z-direction. Many components required and need to be secured to the prototype for independent movement as seen in Figure 7.
- A more complex PD control algorithm, for autonomous hover, is tested to further validate the simulations. Data is collected as the Quad axis prototype is hovering using inflight data logged into a CSV file and shown in below plots for Angle and Angular Velocity data.

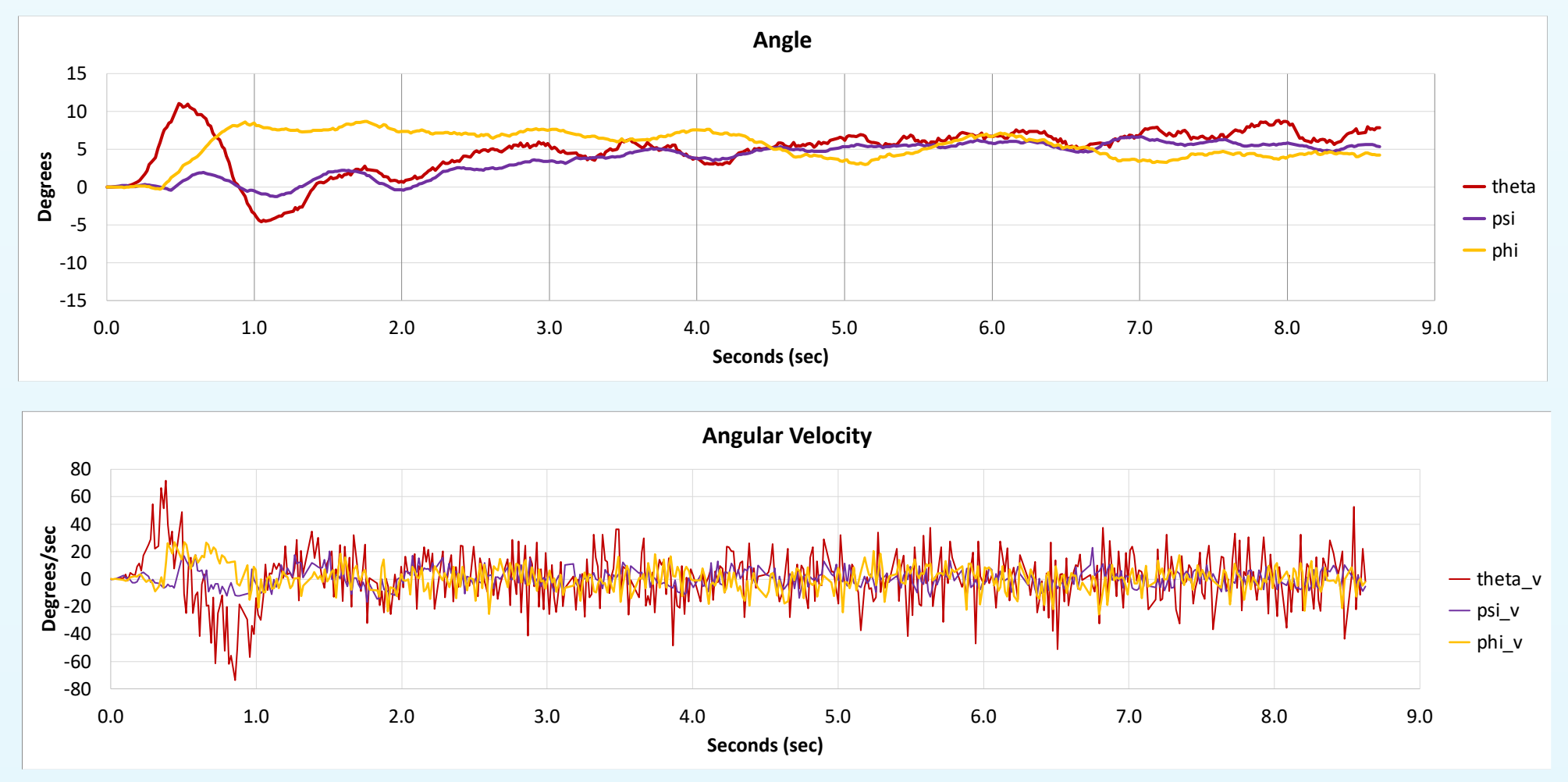


Figure 13. Quad Axis Angle and Angular Velocity Data during Flight

- The more complex control algorithm correcting for the 4 states of the prototype further validates the MATLAB simulations: the Angular Velocity sits very close to zero, while the angles has a slight offset during hover due to steady state error (imbalance).

Conclusion

- Single Loop Control Systems for Quadcopters are more likely to be unstable while navigating in x-,y-direction because the Control system is over-actuated, six degrees of freedom {x, y, z, ϕ, θ, ψ} controlled by only 4 inputs.
- A novel Dual Loop Control System was developed in this project to autonomously generate stability, during hover and navigation. It has 2 independent loops, navigation (every 10th cycle) and hover (every cycle), which work well and are robust since each loop of the algorithm controls 4 states or less with 4 inputs.
- The Dual loop system allows the quadcopter’s hover conditions to be stable between each update of x- y- navigation step, allowing all the control system to hold.
- Four different Control Algorithms were simulated and optimized in MATLAB for hover and navigation dynamics with the novel Dual Loop Methodology to determine what is the best control algorithm. A cost function determined by the settling time, noise input, and maximum range identified best algorithm (Eq 3).

$$Cost\ Function = \sum_{States} \frac{Settling\ time}{5} + \sum_{States} Noise + \sum_{Angles} \frac{Max\ Angle}{50} \tag{3}$$

	Cost Function
PD	12.07
PID	16.55
BSC	25.18
SMC	9.44

- The Sliding Method Control (SMC) performs the best due to its fast settling for angles back to stable position and the reasonable levels on angular velocity required to move the quadcopter in the x-y position. The cost function shows SMC having 9.4 versus 25 for BSC, and 12 and 16 for the PD and PID algorithms. The SMC algorithm can also handle more noise and and steady-state imbalance compared to the other algorithms.
- The single axis prototype generated real world data, which validated the MATLAB simulation. The quad axis prototype, which tested four degrees of motion {z, ϕ, θ, ψ}, further validated the MATLAB simulations of the PD control algorithm. These prototypes demonstrated that imbalance, noise, and disturbance are critical for stabilization in real-world circumstances and affect optimization.

Future Work

- The simulations include several real-world nonidealities, such as noise from sensor measurement, disturbances such as wind, and steady-state errors such as uneven weight distribution and motor gain. An additional nonideality that can be added are limits on maximum angular velocity of the quadcopter.
- The next steps for the Quad axis prototype is to try the remaining control algorithms explored in simulations.
- Once the optimum control algorithm is validated on 4 state prototype, a dual loop control architecture on free running quadcopter with 6 degrees of freedom will be implemented.