THE UNIVERSITY OF ARIZONA

Hands-On Learning with the Low-Cost Bi-Copter: From Fundamentals to Advanced Controls

60698830

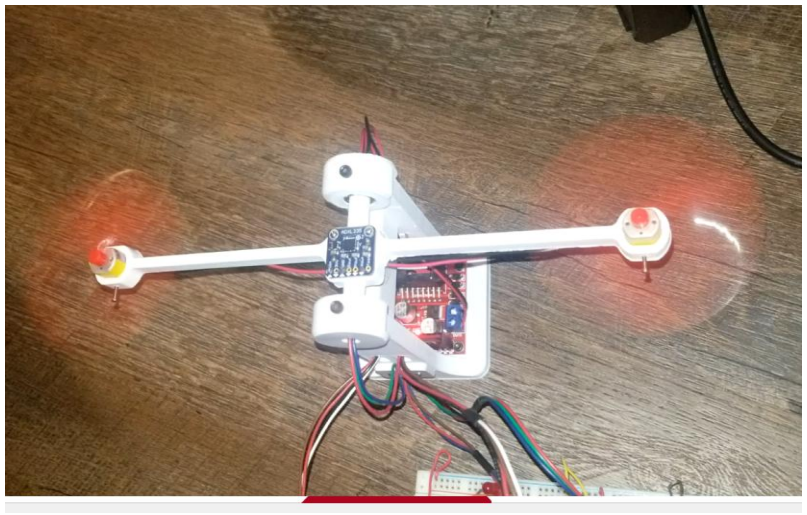
Prof. Eniko T. Enikov,
University of Arizona

Dr. Melda Ulusoy, MathWorks

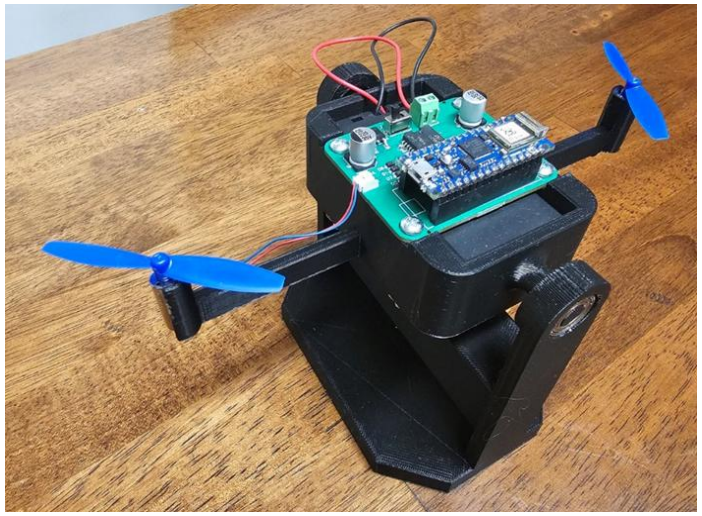
MATLAB EXPO

Motivation and Background: Low-cost hardware, allowing classical and advanced control experiments

Aeropendulum
2011
(USB cable)

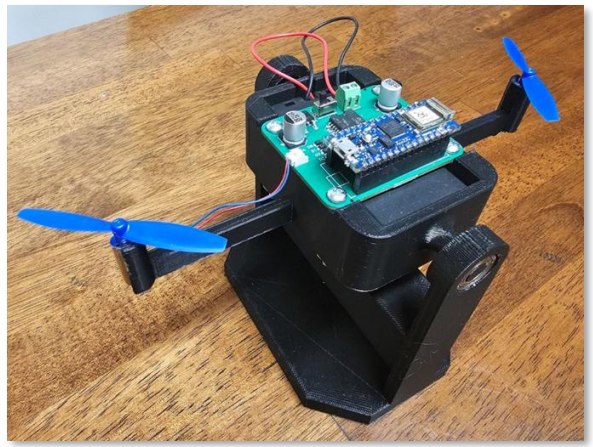


Bi-Copter with
PIC16F690
2020
(USB cable)

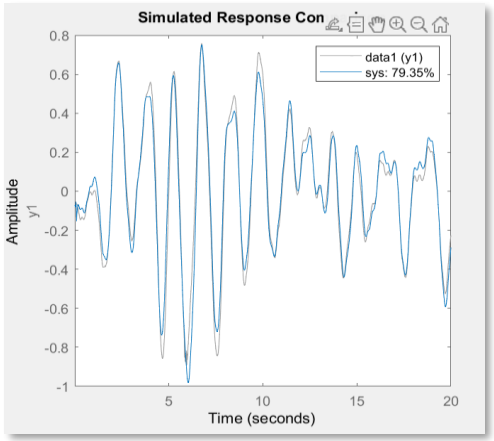


Bi-Copter with
Arduino Nano
2024
(wireless)

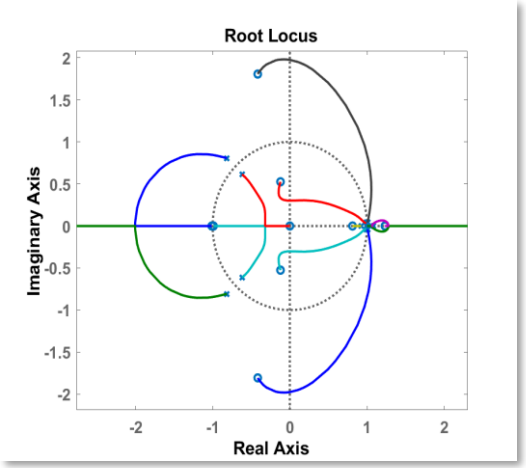
This presentation focuses on the following topics



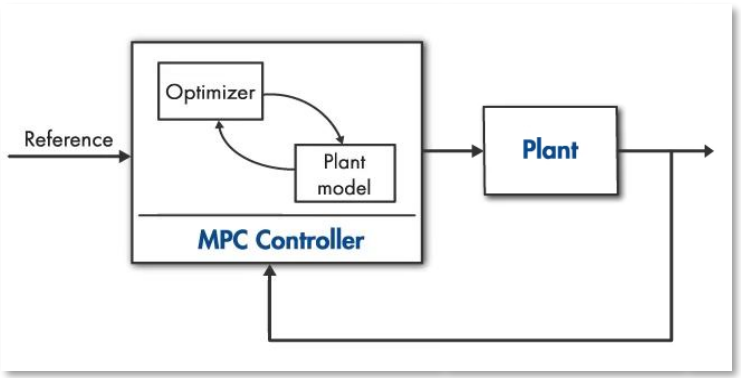
Bi-Copter



Modeling and System Identification



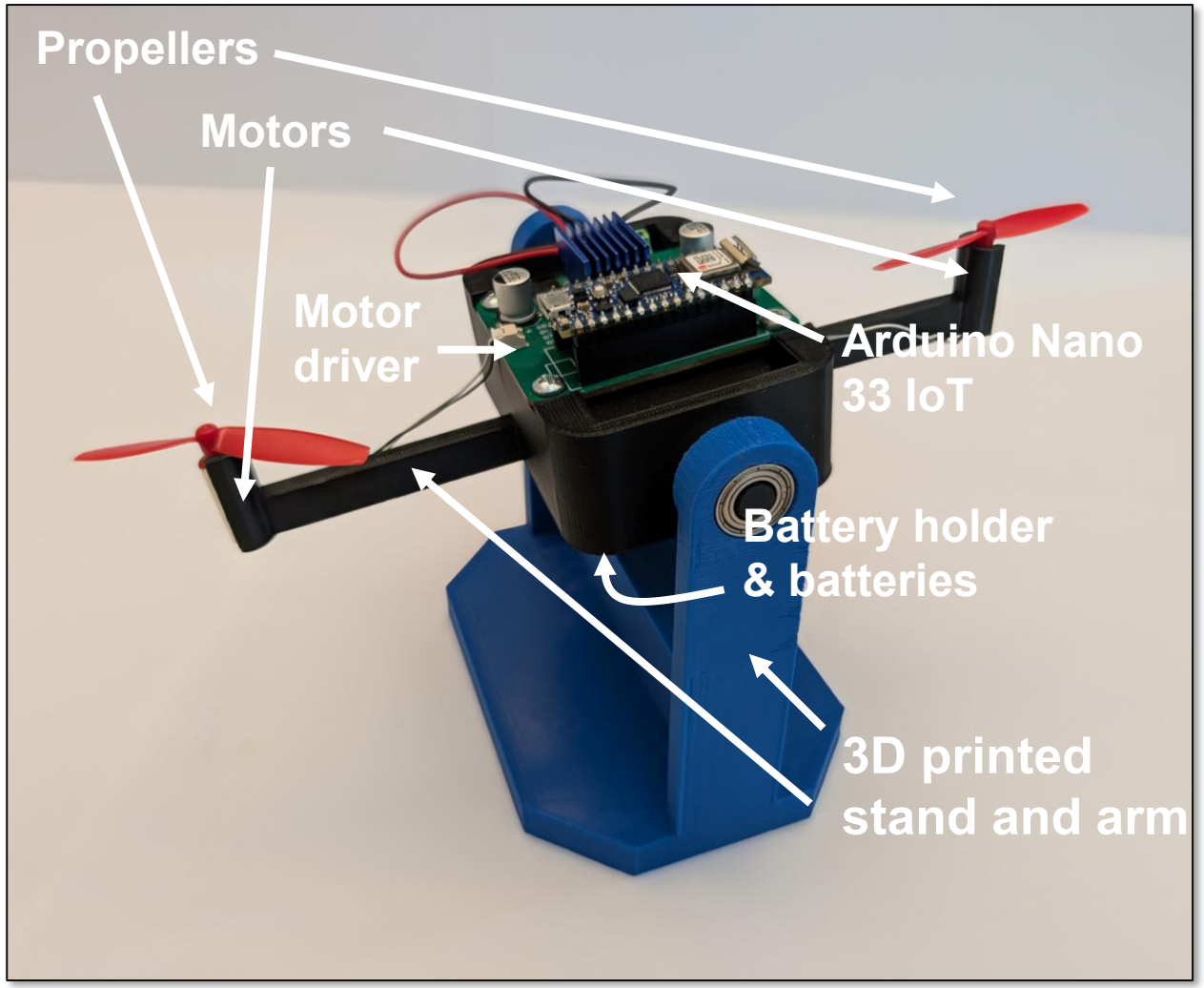
Classical Controller Design

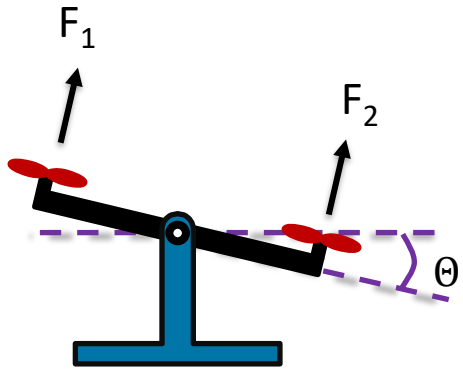


Modern (MPC) Controller Design

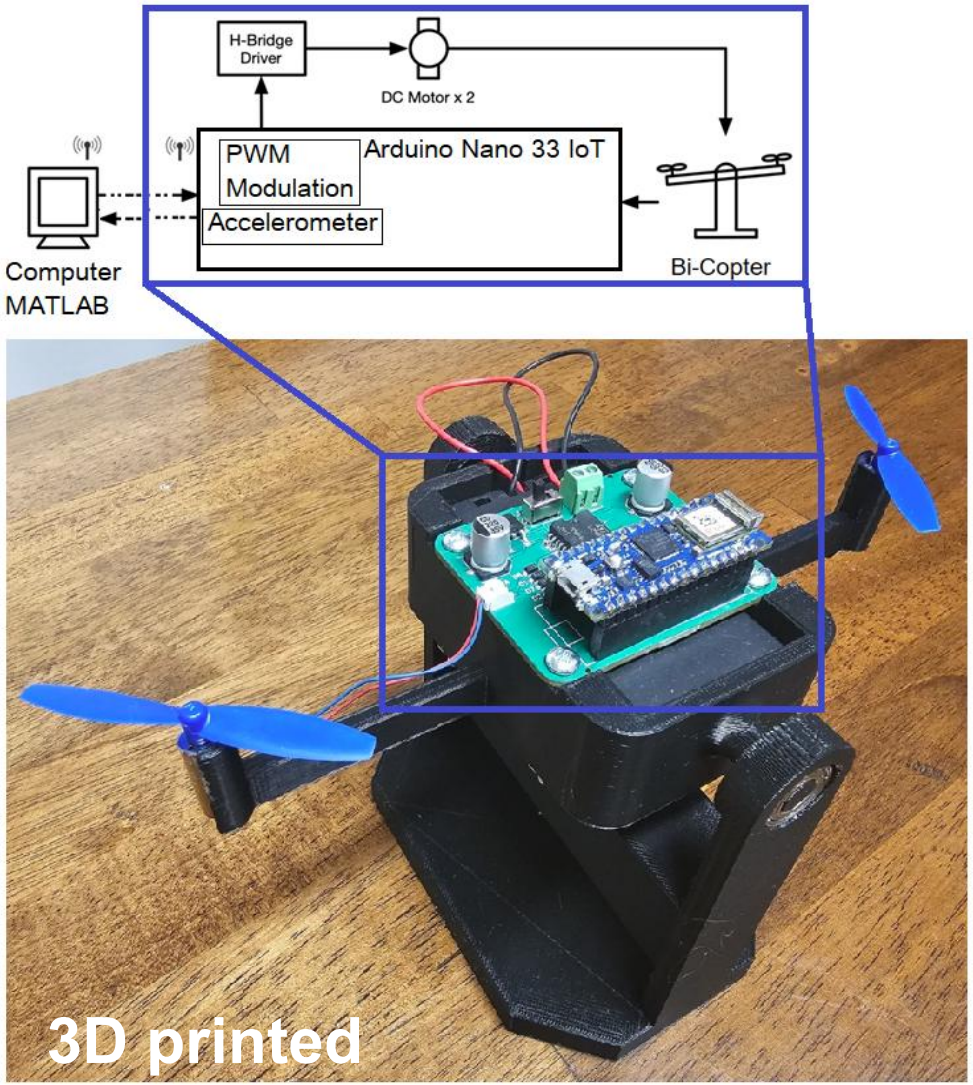
Hardware overview

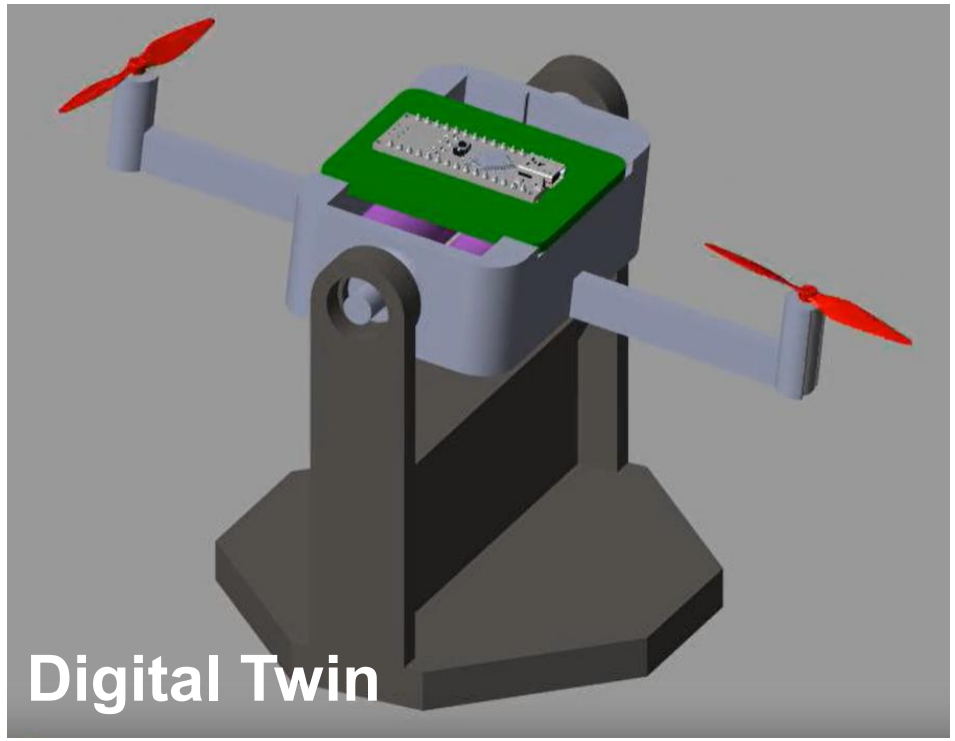
Wireless operation





The low-cost bi-copter and its features



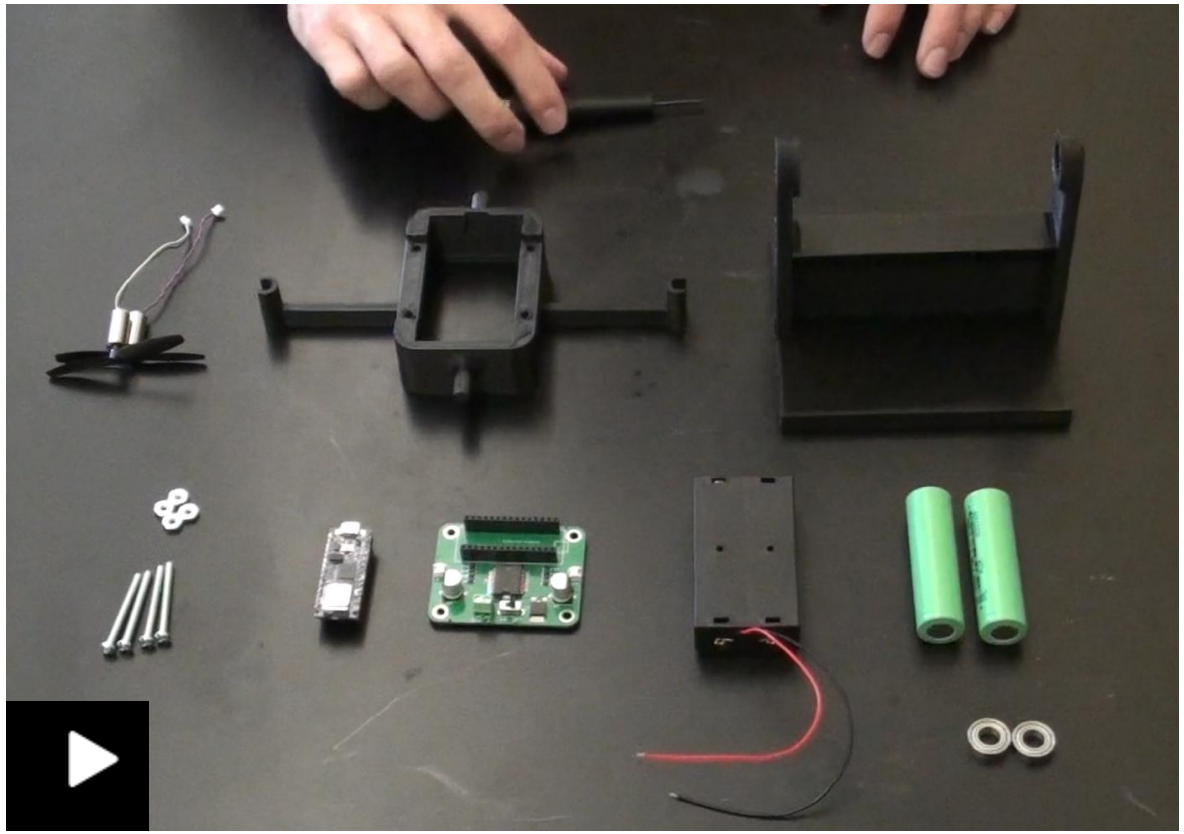


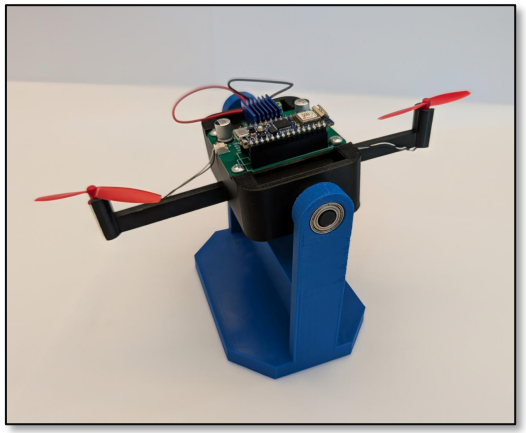
3D printing and assembly instructions are available for the bi-copter hardware

Assembly instructions [link]

3D printing instructions [link]

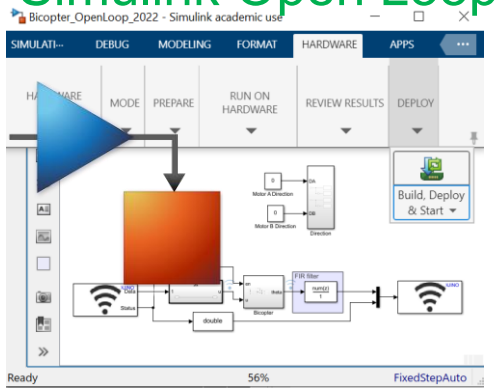
Source Files [link]



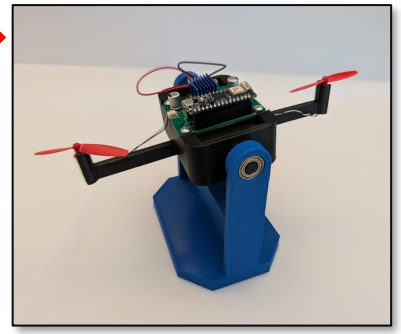


Bi-copter runs a pre-loaded Simulink controller (open- or closed-loop) and sends/receives data to MATLAB Live Script

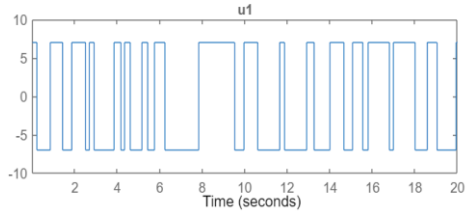
Simulink Open Loop

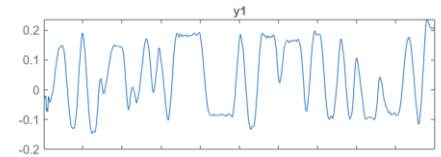


Step 1: Deploy Controller

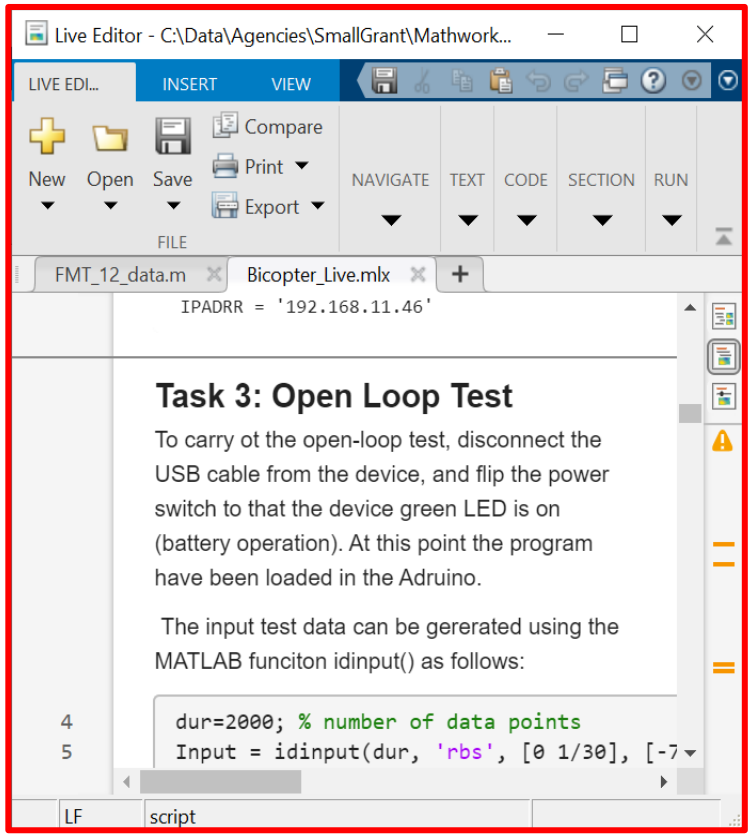


Step 2: Run Live Script

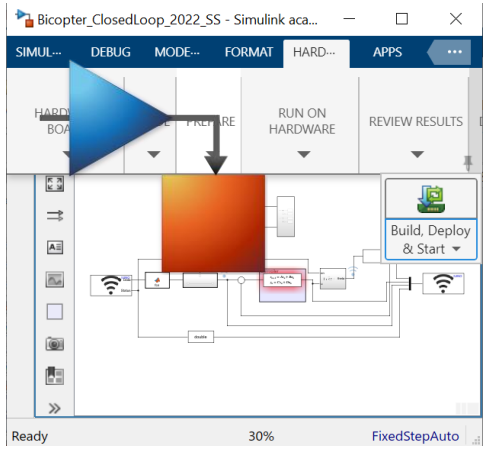




Live Script

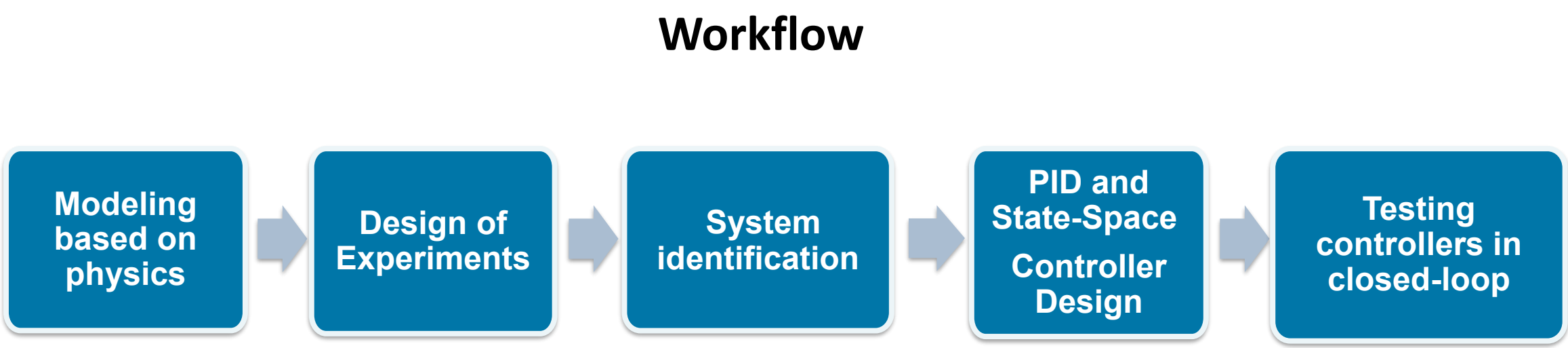


Simulink Closed-Loop

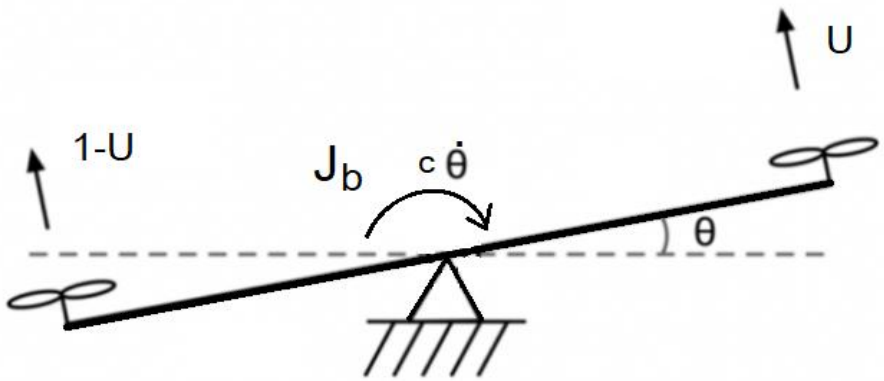


Parameter Updates Via Workspace

Workflow for classical controller design



Modeling based on physics (linear approximation)



$$\text{Propeller: } J_m\frac{d\omega_m}{dt} + b\omega_m = K_m i \tag{1}$$

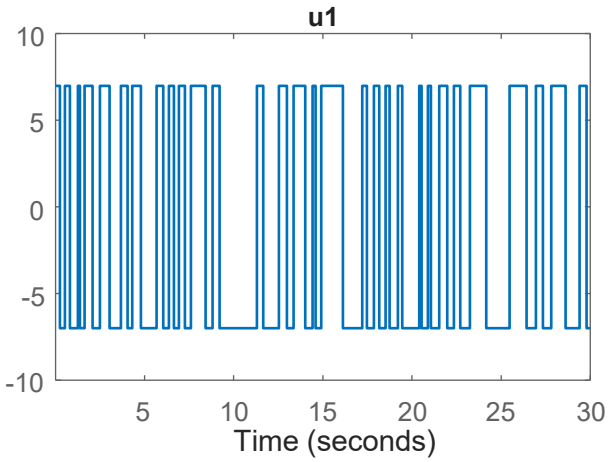
$$\text{DC Motors: } L\frac{di}{dt} + Ri + K_b\omega_m = v, \tag{2}$$

$$\frac{\Omega_m(s)}{V(s)} = \frac{K_m}{(J_m s + b)(Ls + R) + K_b K_m}, \tag{3}$$

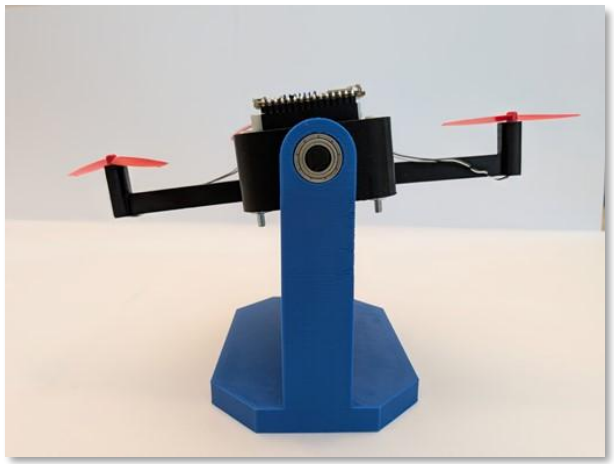
Overall Model:

$$G(s) = \frac{\Theta(s)}{U(s)} = \frac{K_1}{(J_b s^2 + cs + k)[(J_m s + b)(Ls + R) + K_2]}, \tag{4}$$

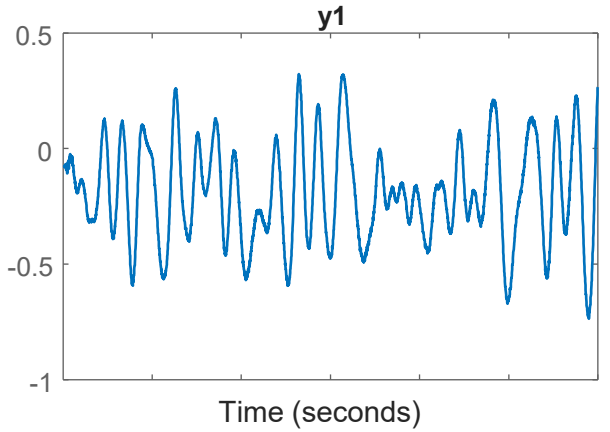
Design of experiments are performed to collect input-output data for system identification



Excite

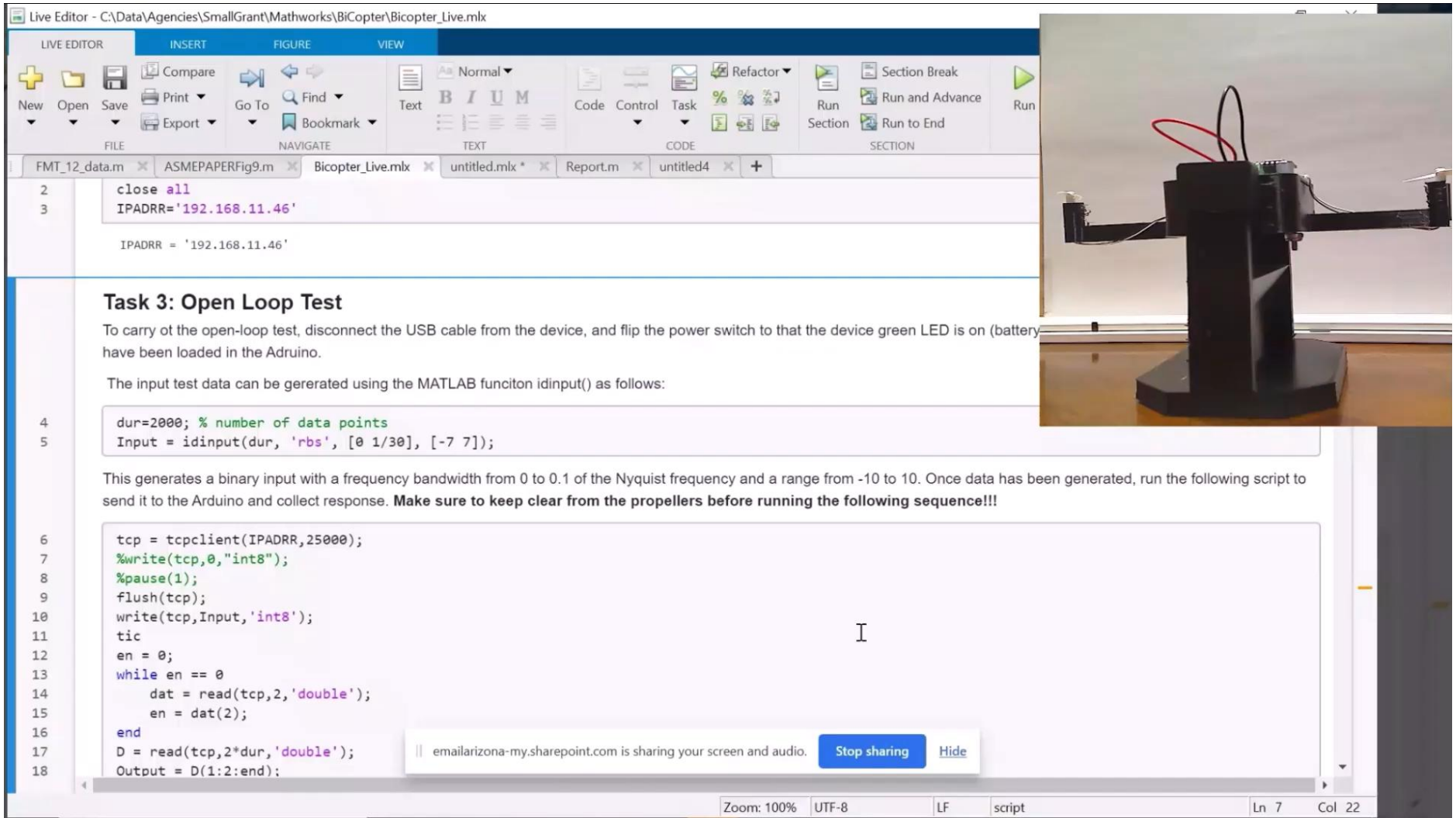


Collect response



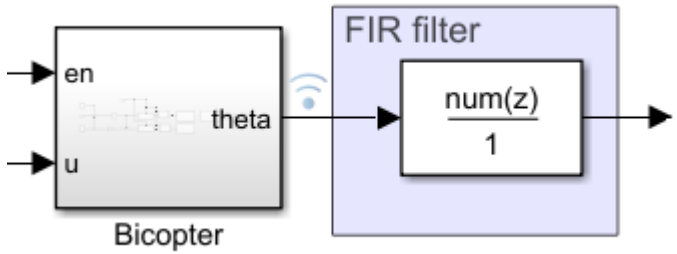
All experiments are carried out using MATLAB Live Script interfacing with the bi-copter

Modeling based on physics → Design of Experiments → **System identification** → PID and State-Space Controller Design → Testing controllers in closed-loop



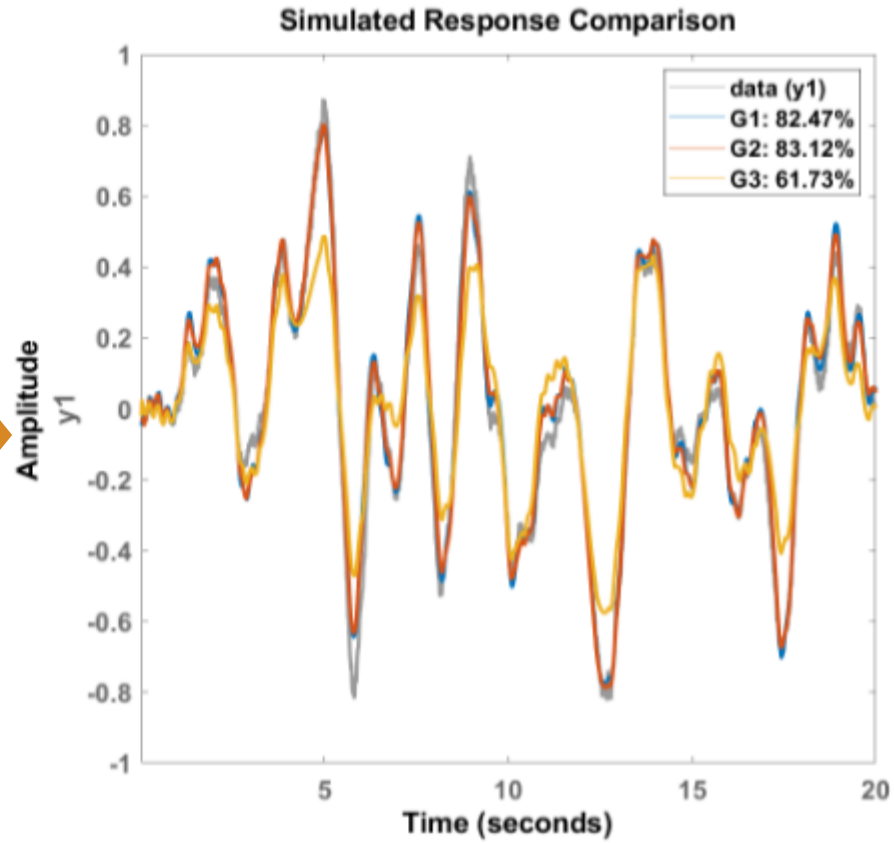
MATLAB EXPO

Select most appropriate model



System ID Models

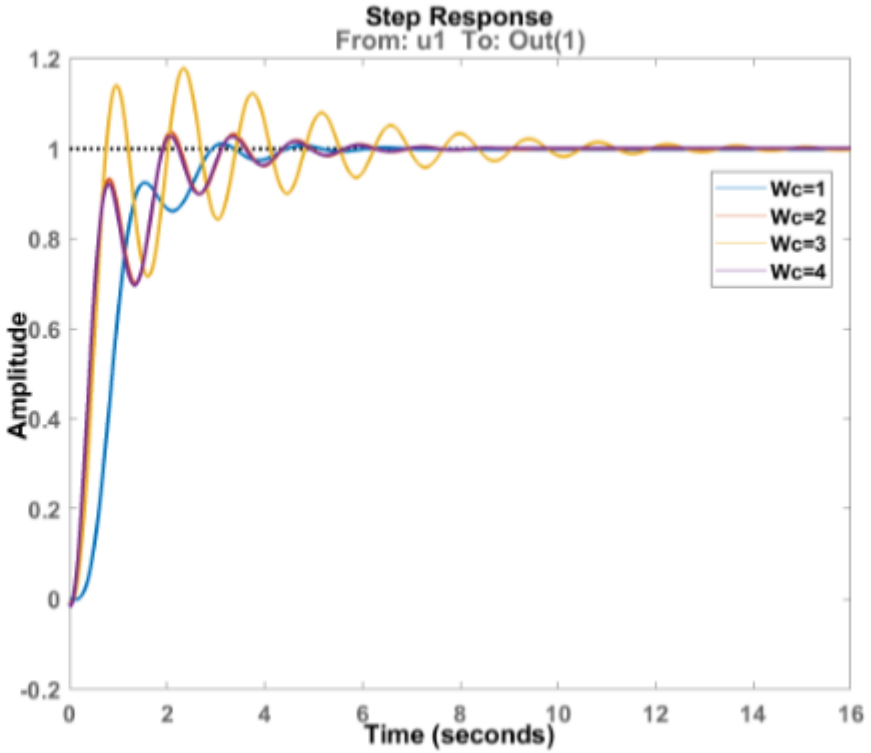
```
G1 = tfest(data,4,3,'Ts',ts)
G2=armax(data,[4 4 10 0])
G3 = n4sid(data,4)
compare(data,G1,G2,G3)
```



PID controller is tuned using different gain cross-over frequencies

PIDtune

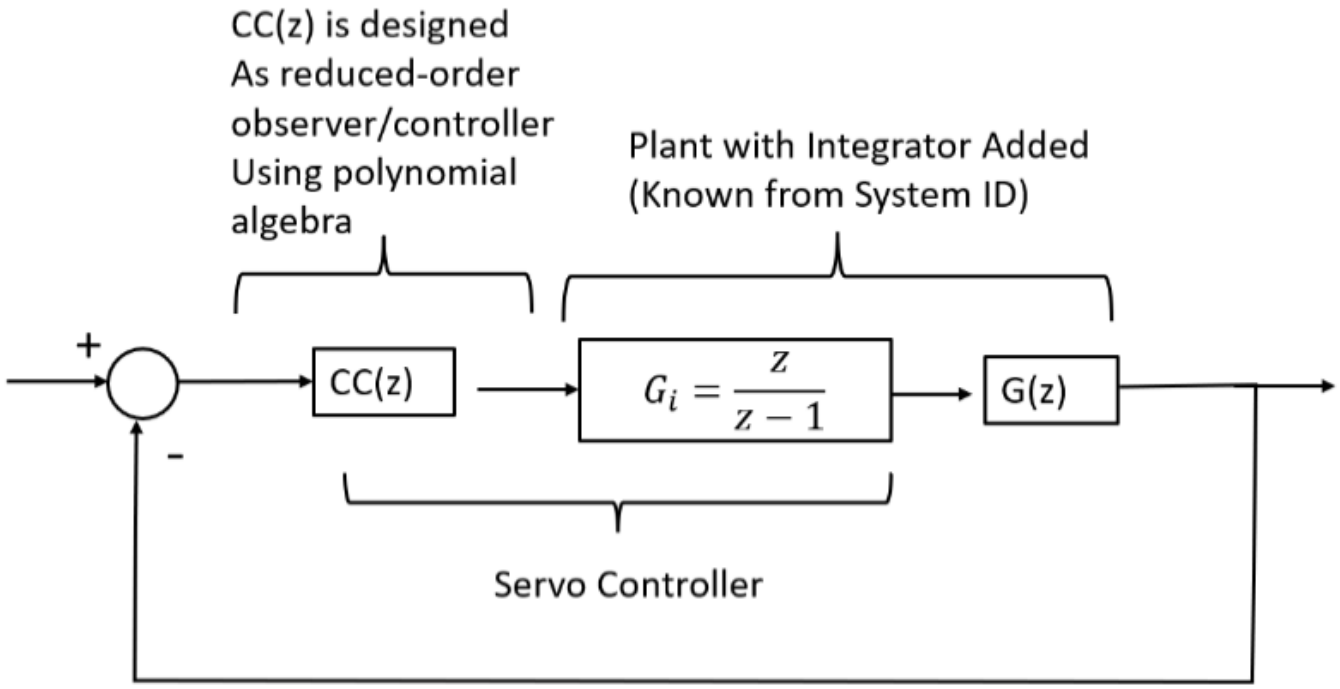
```
C1=pidtune(G2,C0,1)
C2=pidtune(G2,C0,2)
C3=pidtune(G2,C0,3)
C4=pidtune(G2,C0,4)
step(feedback(C1*G2,1),...
feedback(C2*G2,1),...
feedback(C3*G2,1),...
feedback(C4*G2,1))
```




```
                 Ts*z           z-1
C2=  Kp + Ki * ------ + Kd * ------
               z-1            Ts*z
with Kp = -6.79, Ki = -23.1, Kd = -0.499, Ts = 0.01
```

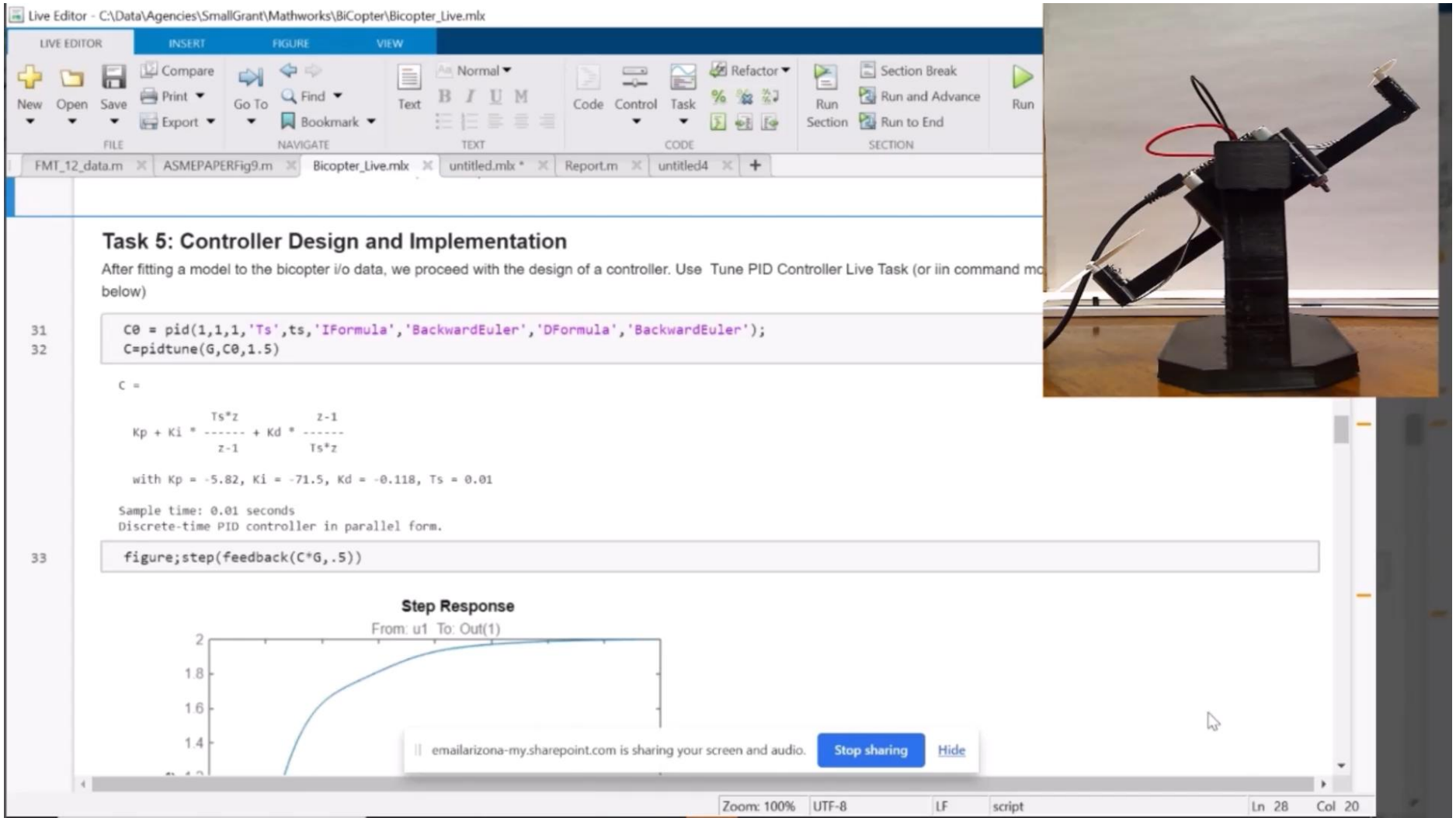
Servo controller is designed using discrete linear quadratic regulator method (dLQR function)

$$\int_0^\infty y^2 + Ru^2 dt = \int_0^\infty x^T C^T C x + Ru^2 dt$$



Optimal pole locations are obtained for various values of the parameter R. Let's see it in practice!

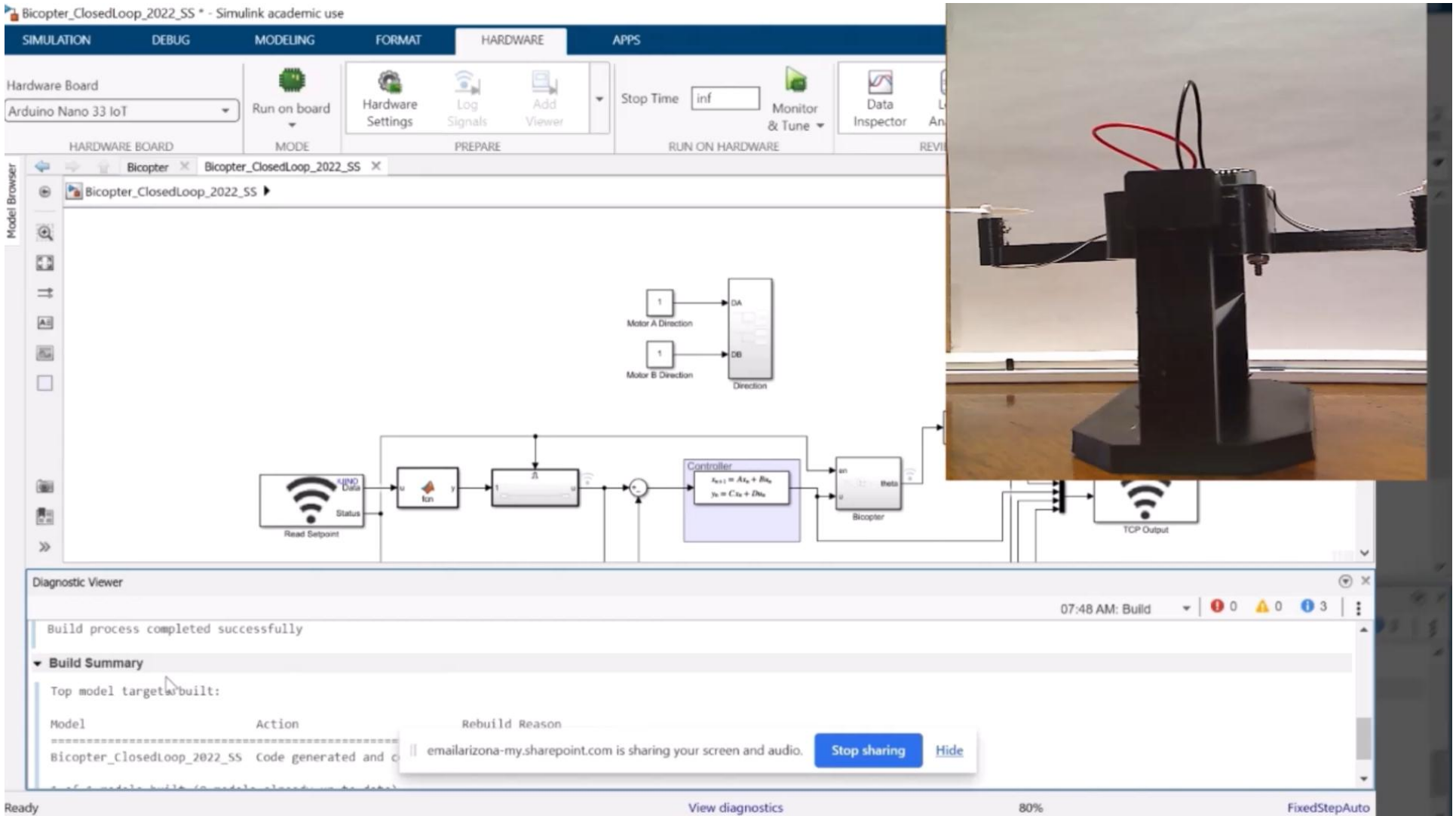
Modeling based on physics → Design of Experiments → System identification → **PID and State-Space Controller Design** → Testing controllers in closed-loop



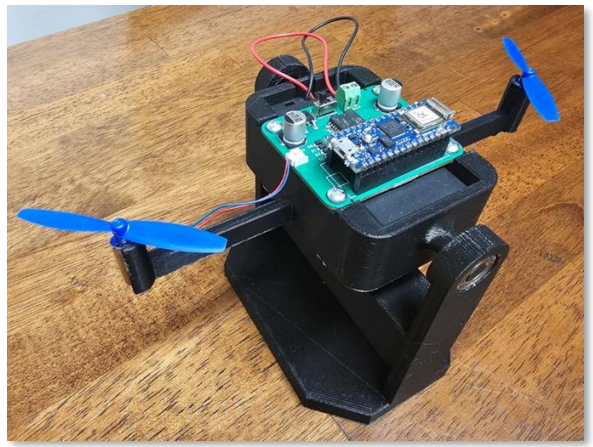
MATLAB EXPO

Closed-loop testing of the servo controller

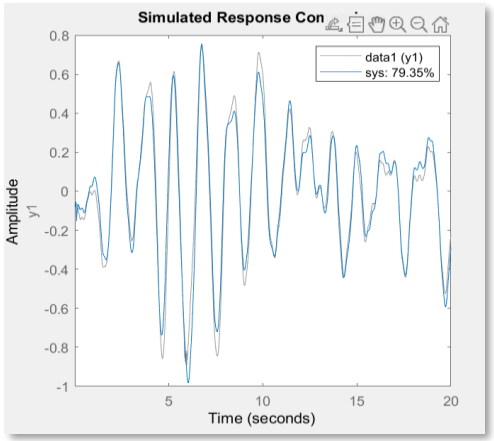
Modeling based on physics → Design of Experiments → System identification → PID and State-Space Controller Design → **Testing controllers in closed-loop**



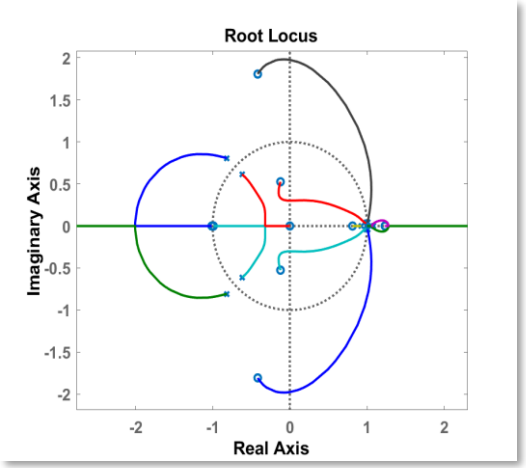
Next, we will focus on MPC design and deployment



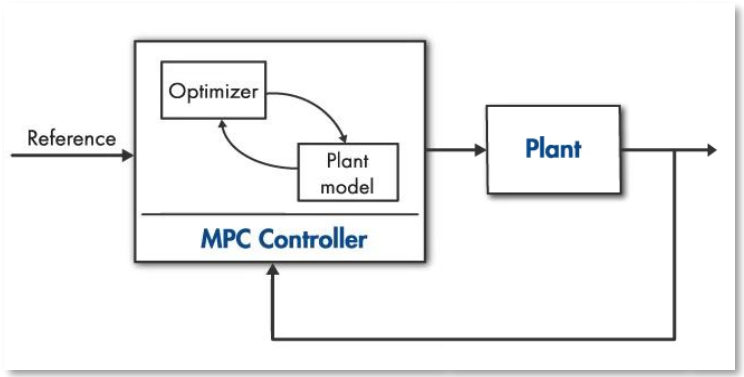
Bi-Copter



Modeling and System Identification

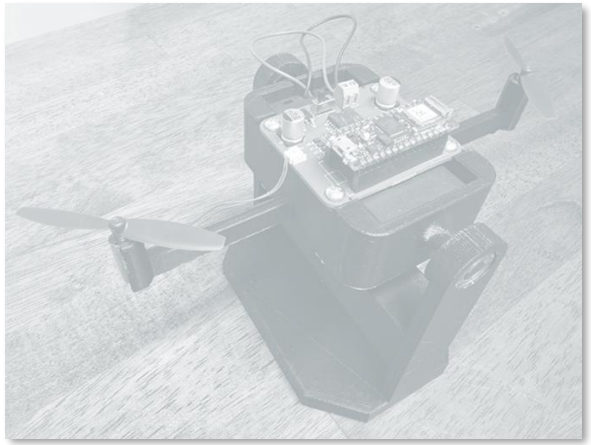


Classical Controller Design

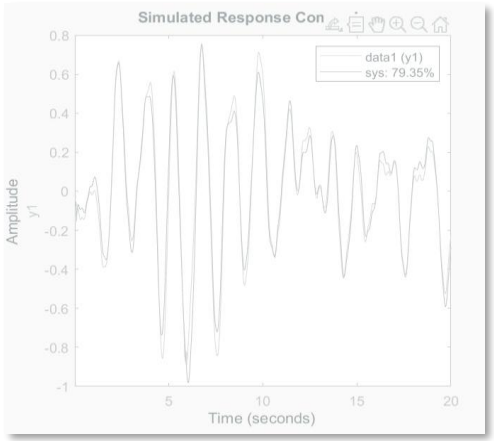


Modern (MPC) Controller Design

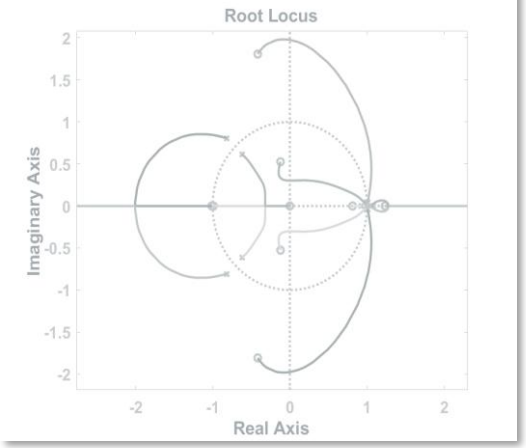
Next, we will focus on MPC design and deployment



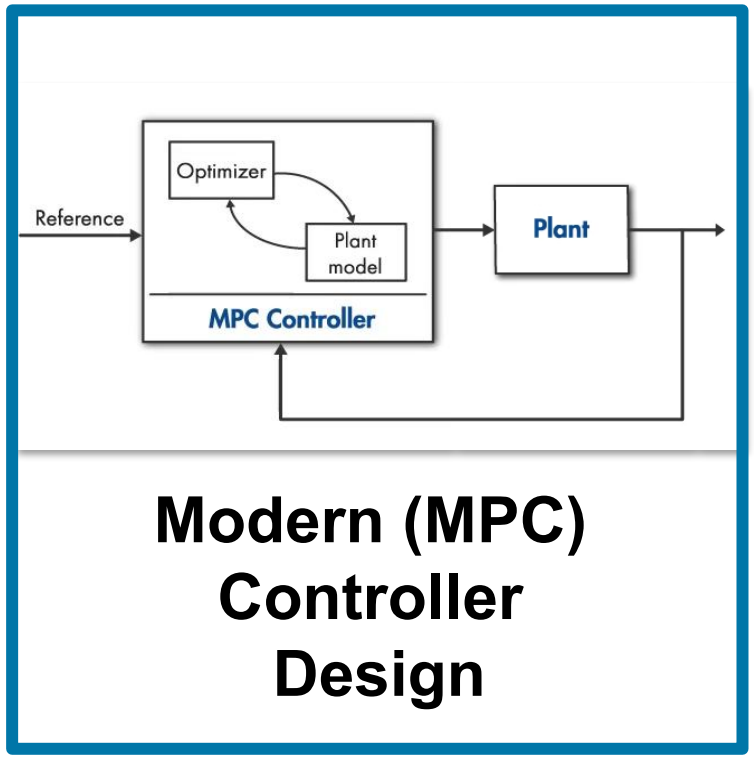
Bi-Copter



Modeling and System Identification

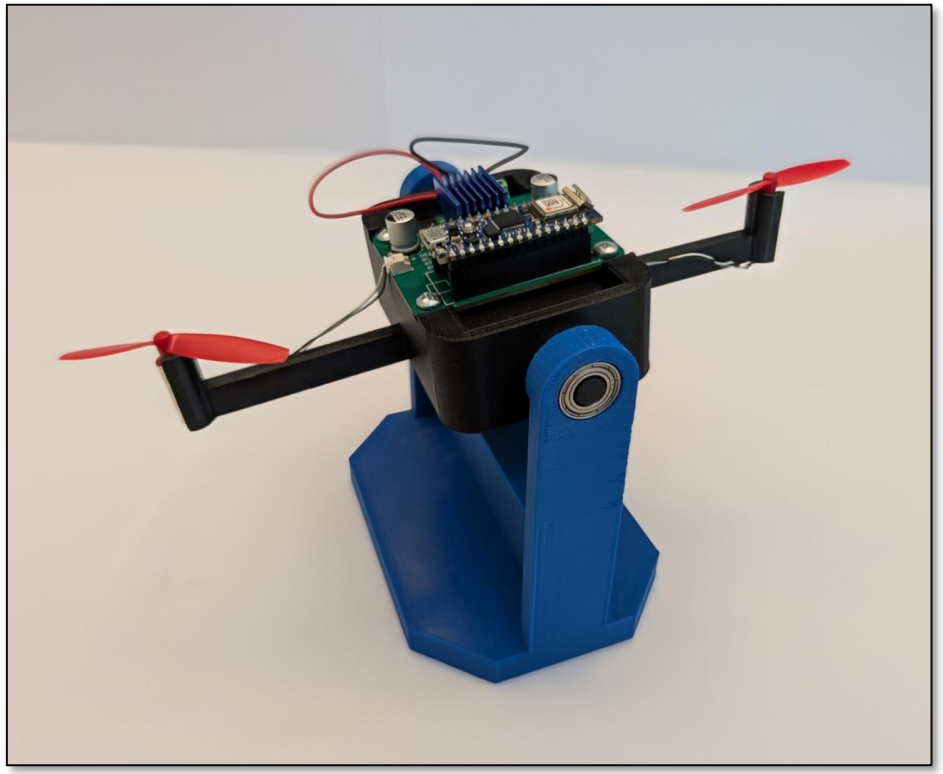


Classical Controller Design



Modern (MPC) Controller Design

In addition to fundamentals, bi-copter hardware also lets educators introduce advanced control concepts such as MPC.



Controller design and deployment

System identification

PID control

Algorithm development

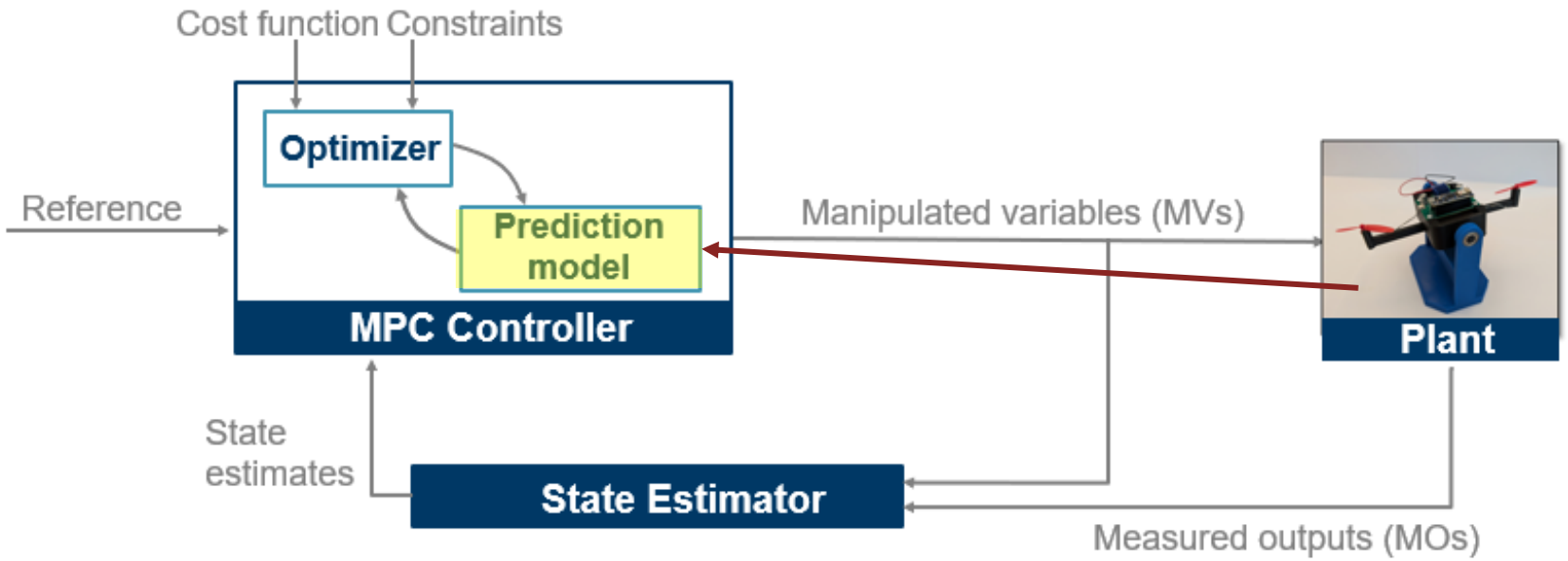
Model predictive control (MPC)

Mechatronics

Hands-on learning

Sensors and actuators

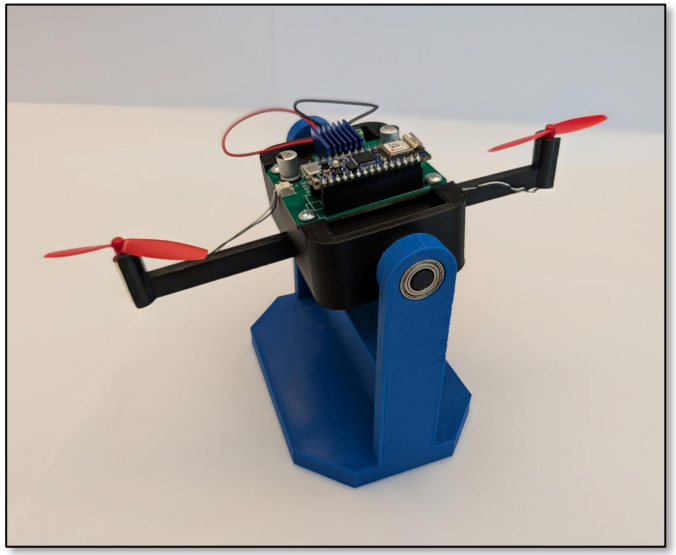
MPC uses a dynamic model of a system to predict future behavior and optimize control actions in real time, subject to constraints.



Here’s the end-to-end workflow for MPC design and deployment.

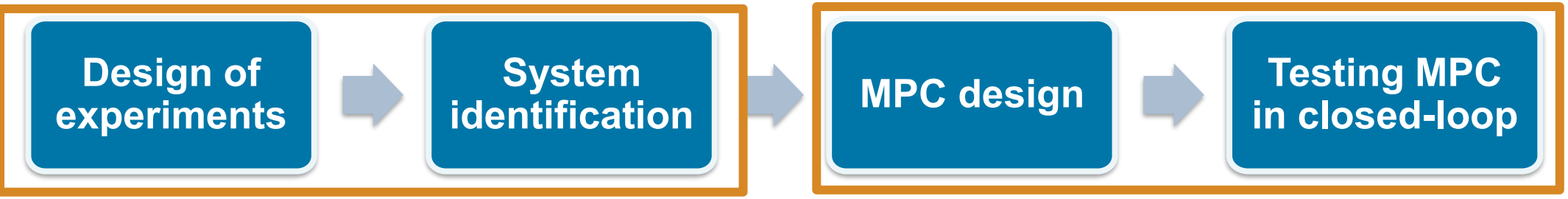
Workflow

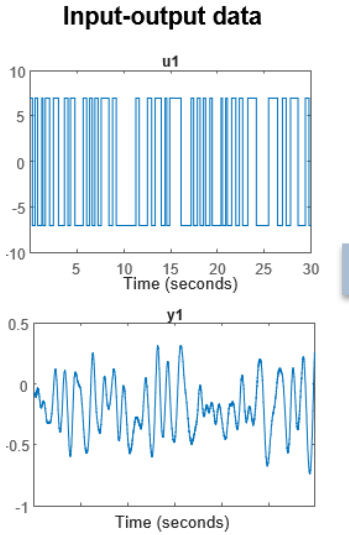
Design of experiments → **System identification** → **MPC design** → **Testing MPC in closed-loop**

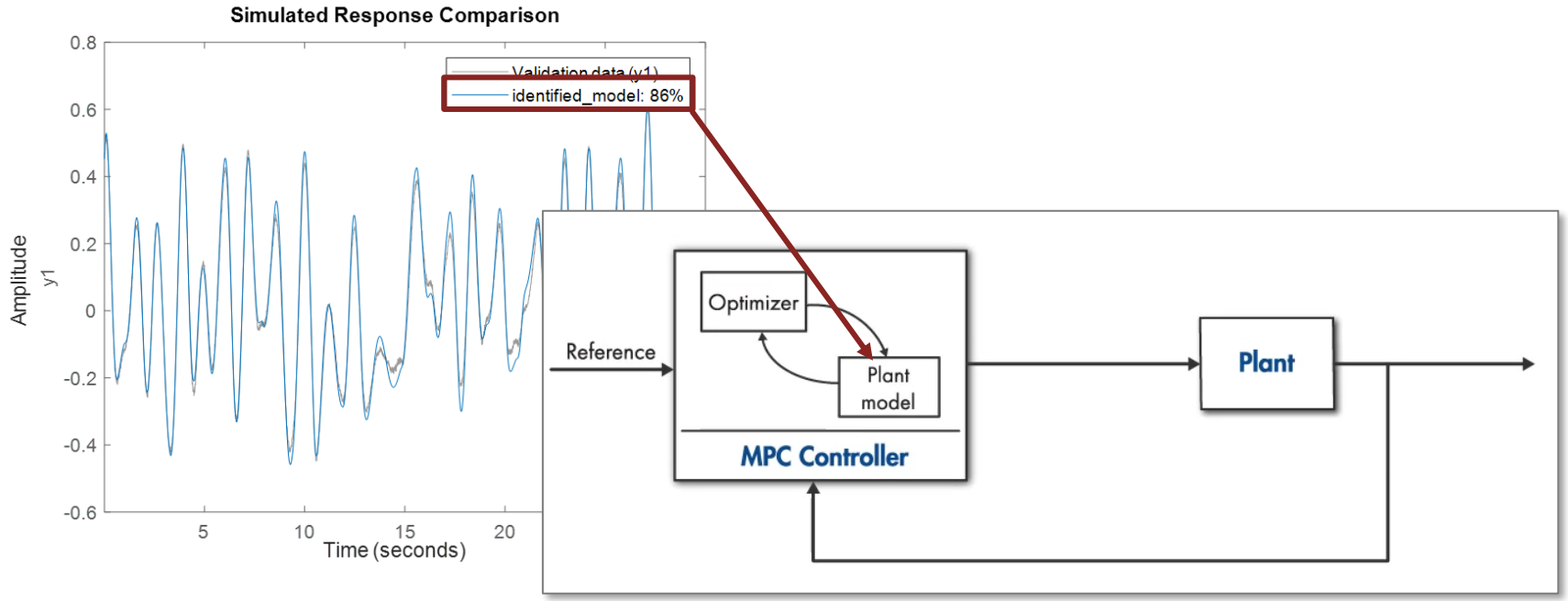


Here's the end-to-end workflow for MPC design and deployment.

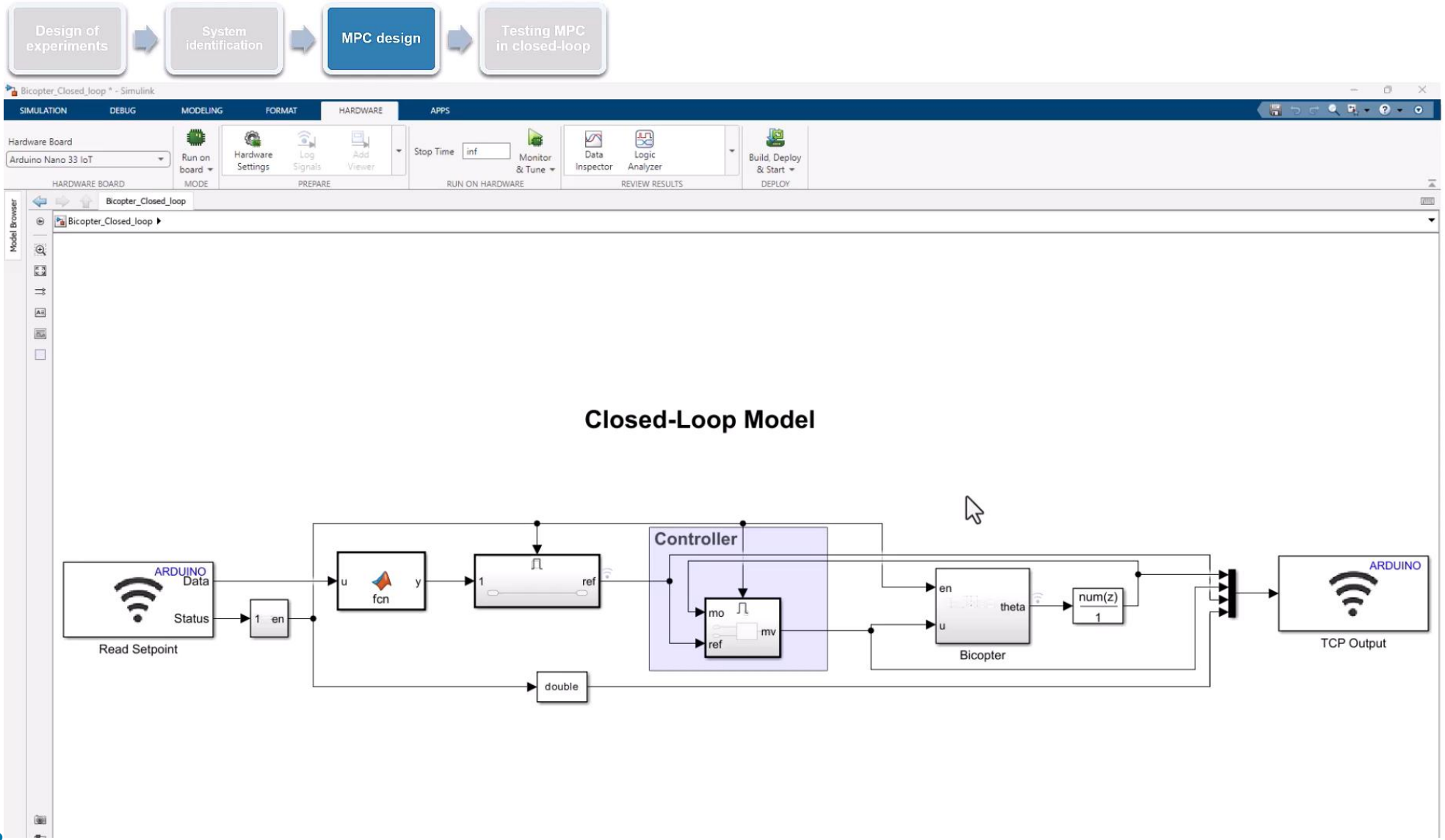
Workflow







An MPC is designed using the MPC Designer app to maintain the bi-copter at the specified reference angles.



Testing the deployed MPC in closed-loop

C:\MPC\Bi_copter_MPC.mlx *

LIVE EDITOR | INSERT | FIGURE | VIEW

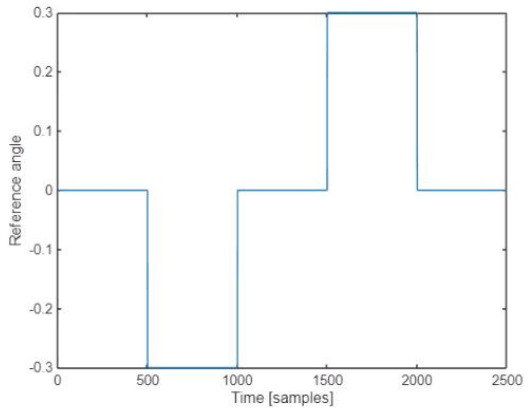
New | Open | Save | Compare | Print | Export | Go To | Find | Bookmark | Text | Normal | Code | Control | Task | Refactor | Run Section | Section Break | Run and Advance | Run to End | Run | Step | Stop

FILE | NAVIGATE | TEXT | CODE | SECTION | RUN

Testing the designed MPC in closed-loop

For testing the designed MPC, deploy the closed-loop model to Arduino first.

```
% Generate three-level square wave reference signal
length = 2500;
Ref = zeros(length,1);
Ref(501:1000) = -0.30;
Ref(1501:2000) = 0.30;
plot(Ref);
xlabel("Time [samples]");ylabel("Reference angle");
```

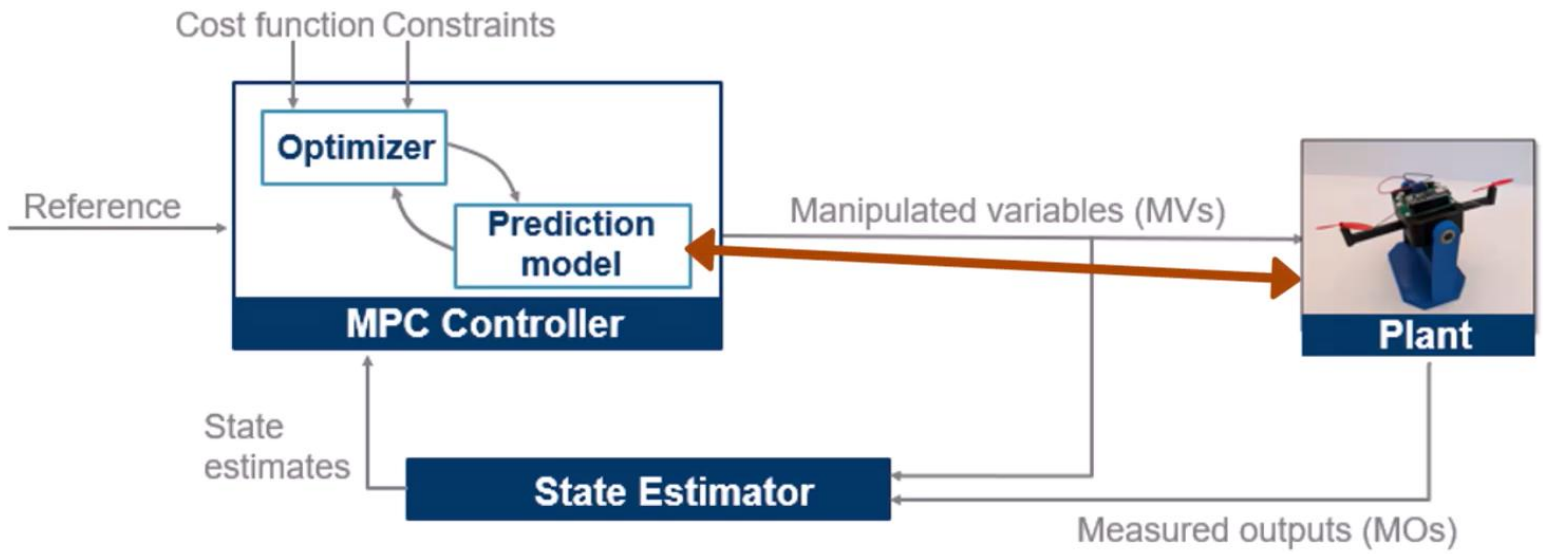



```
% Run experiment to test the designed MPC in closed-loop
IPADRR = '192.168.1.27'; %typically IP address remains unchanged after deployment of closed-loop model
clear tcp;
tcp = tcpclient(IPADRR,25000);
write(tcp,0,'single');
pause(5);
flush(tcp);
write(tcp,Ref,'single');
```

Zoom: 110% | UTF-8

MATLAB EXPO

MPC successfully compensates for the unmodeled disturbance.



In summary, educators can use the bi-copter hardware to provide students with hands-on experience across a wide range of control topics, from fundamentals to advanced controls.

Check out these resources for more information:

- Prof. Enikov's repository contains:
  - Bi-copter 3D printing and assembly instructions
  - MATLAB code and Simulink models for system identification & PID / state-space control
- MPC with bi-copter
  - MPC code and models
  - Tech talk video

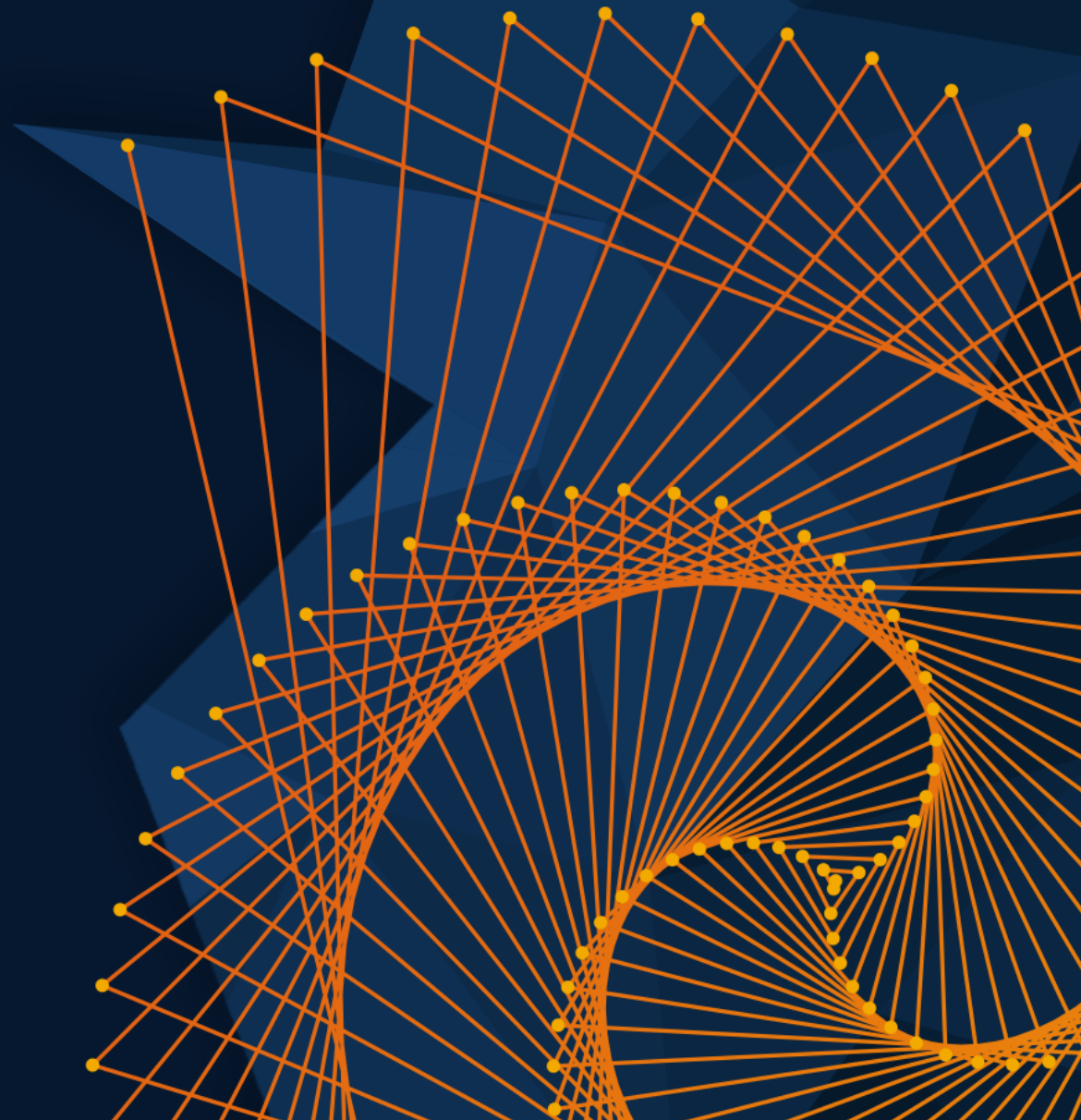
System Identification and MPC Design using a Low-Cost Bi-Copter Hardware

in/eniko-enikov-618b8b106

in/melda-ulusoy

MATLAB EXPO

Thank you



MathWorks®