

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

60698830

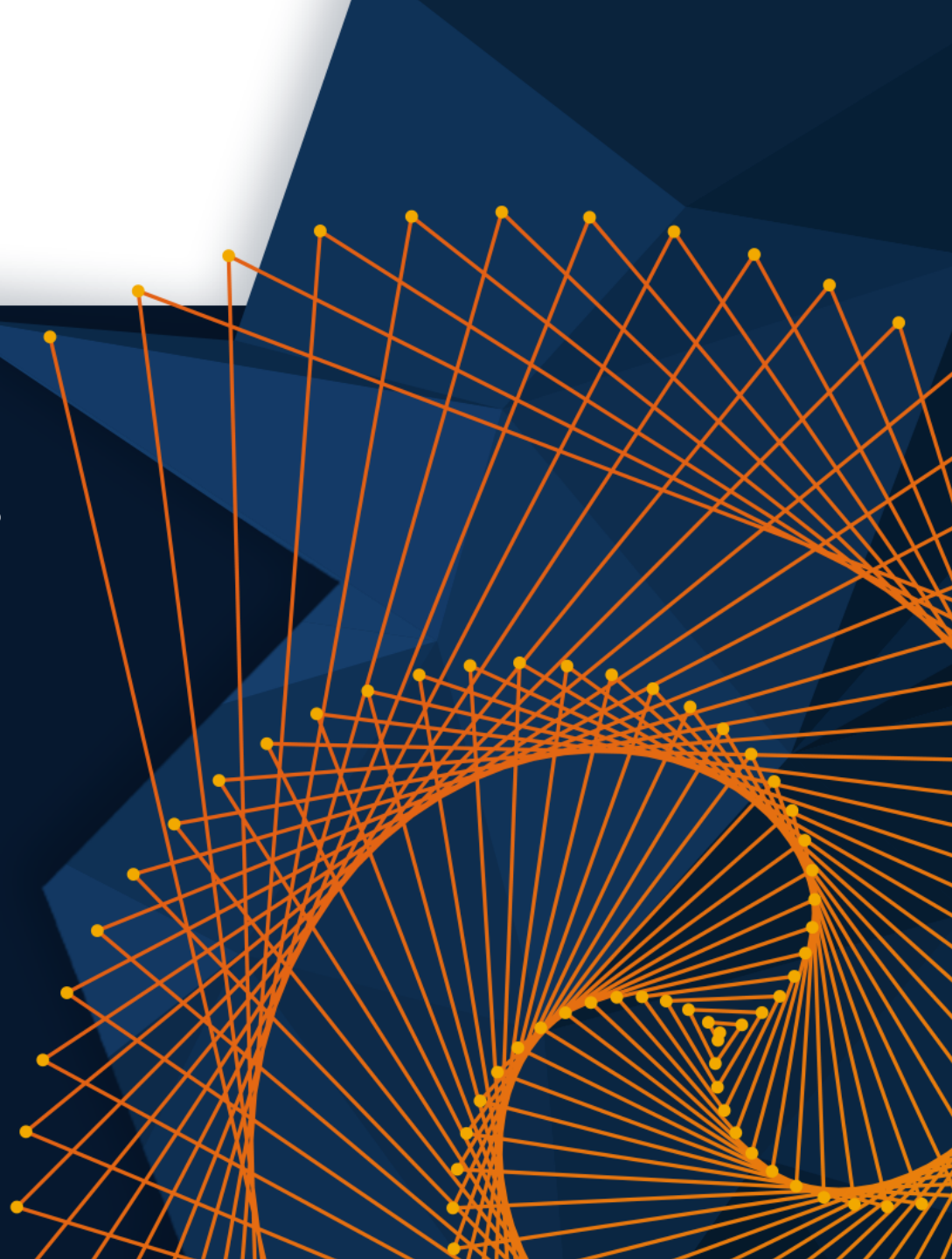
*Prof. Eniko T. Enikov,
University of Arizona*



Dr. Melda Ulusoy, MathWorks



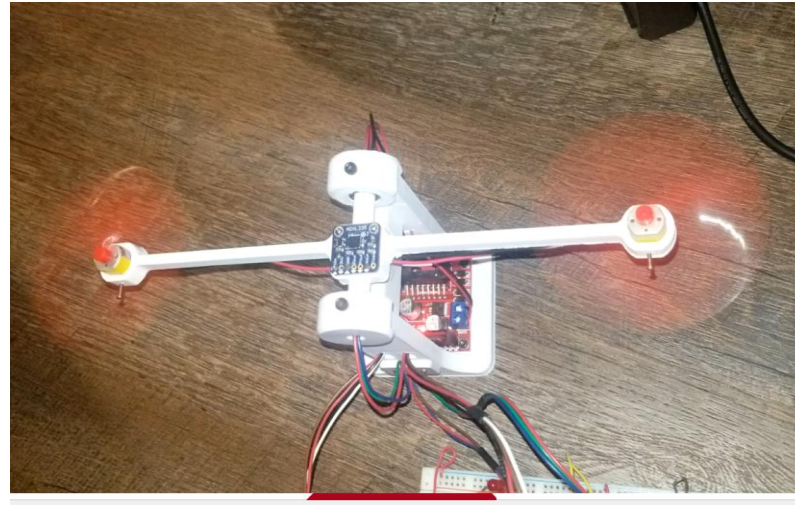
MATLAB EXPO



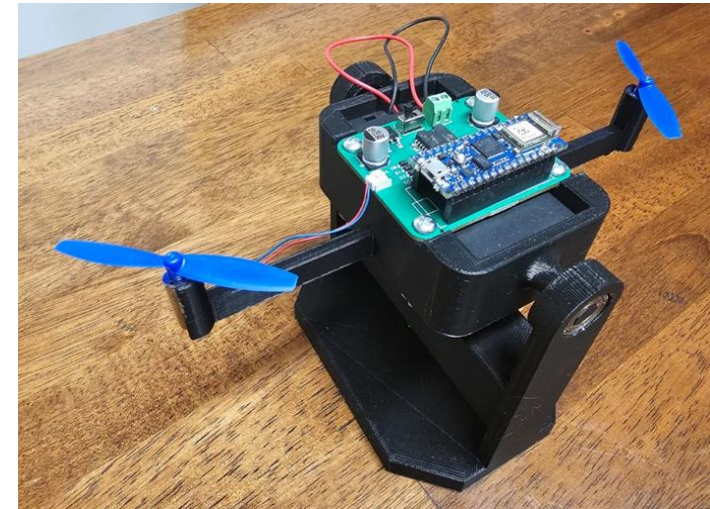
Motivation and Background: Low-cost hardware, allowing classical and advanced control experiments



**Aeropendulum
2011
(USB cable)**

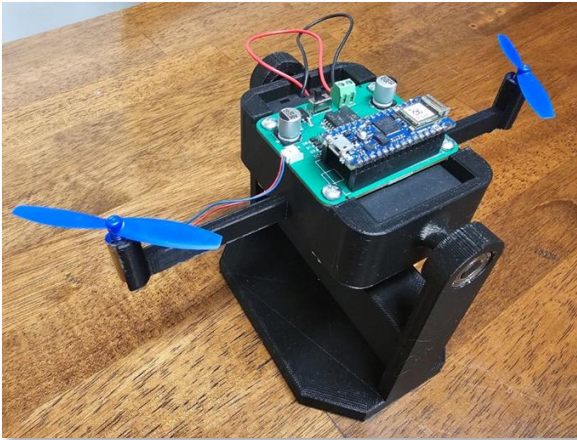


**Bi-Copter with
PIC16F690
2020
(USB cable)**

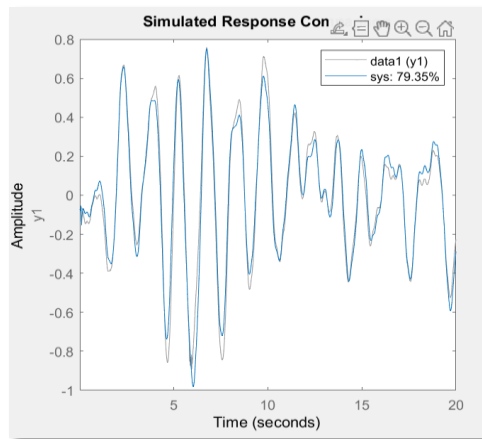


**Bi-Copter with
Arduino Nano
2024
(wireless)**

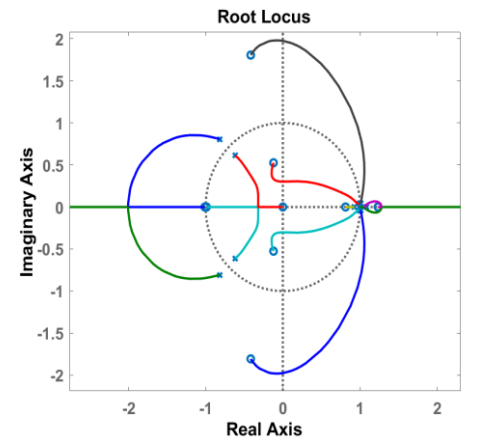
This presentation focuses on the following topics



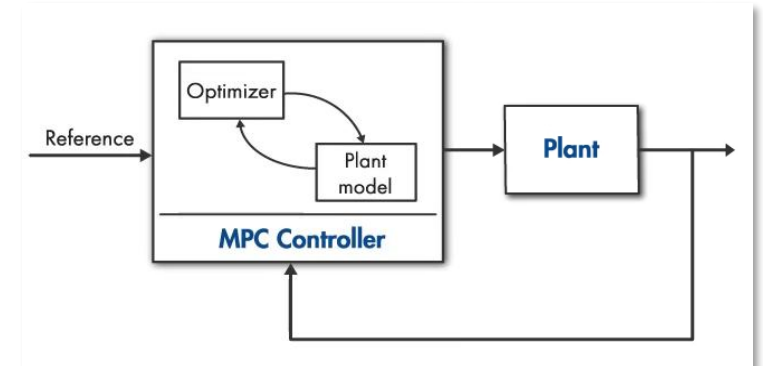
Bi-Copter



**Modeling and
System
Identification**



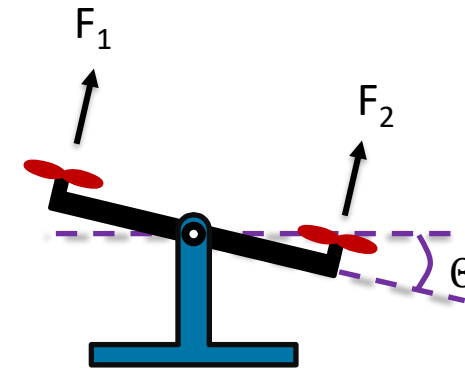
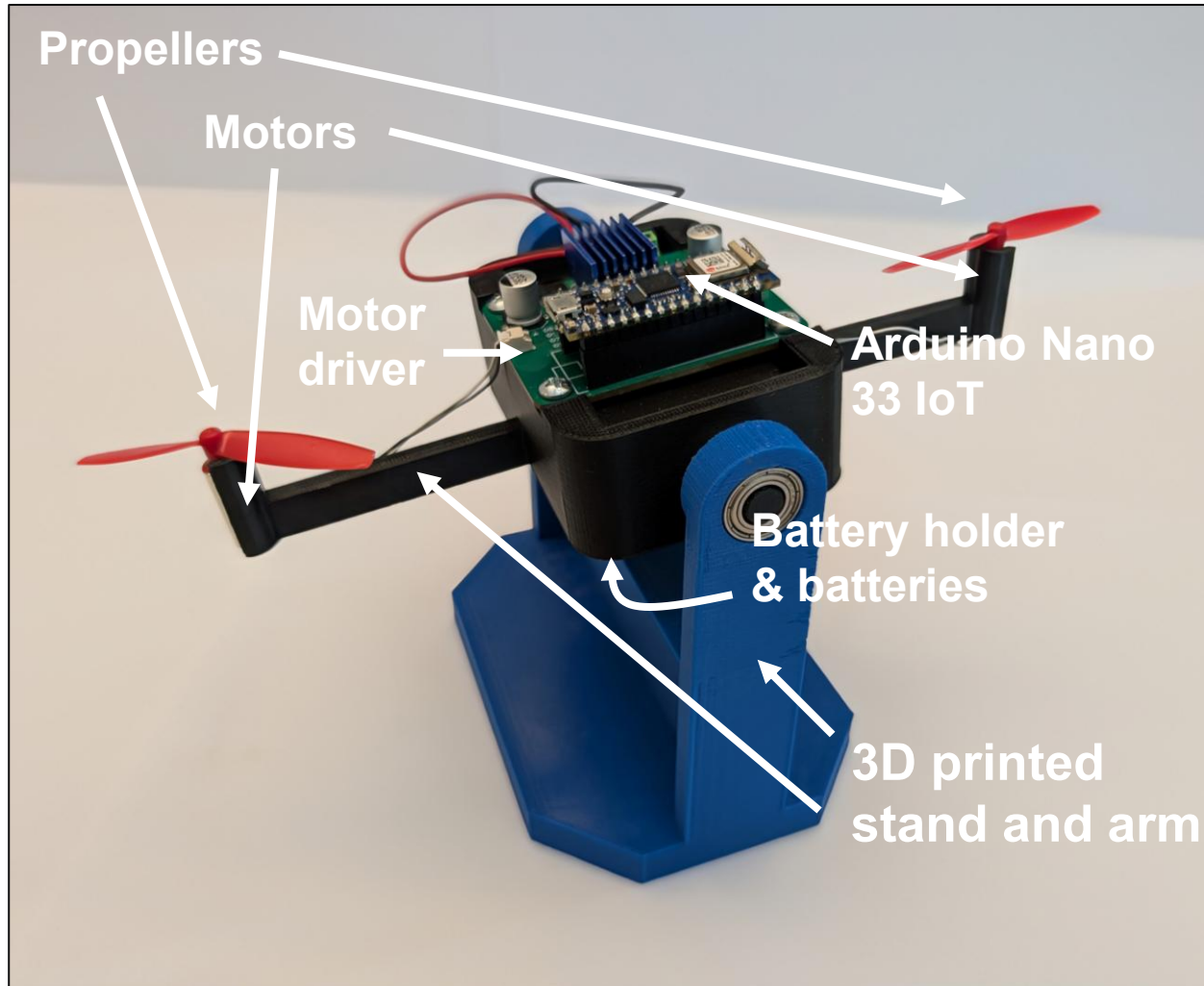
**Classical
Controller
Design**



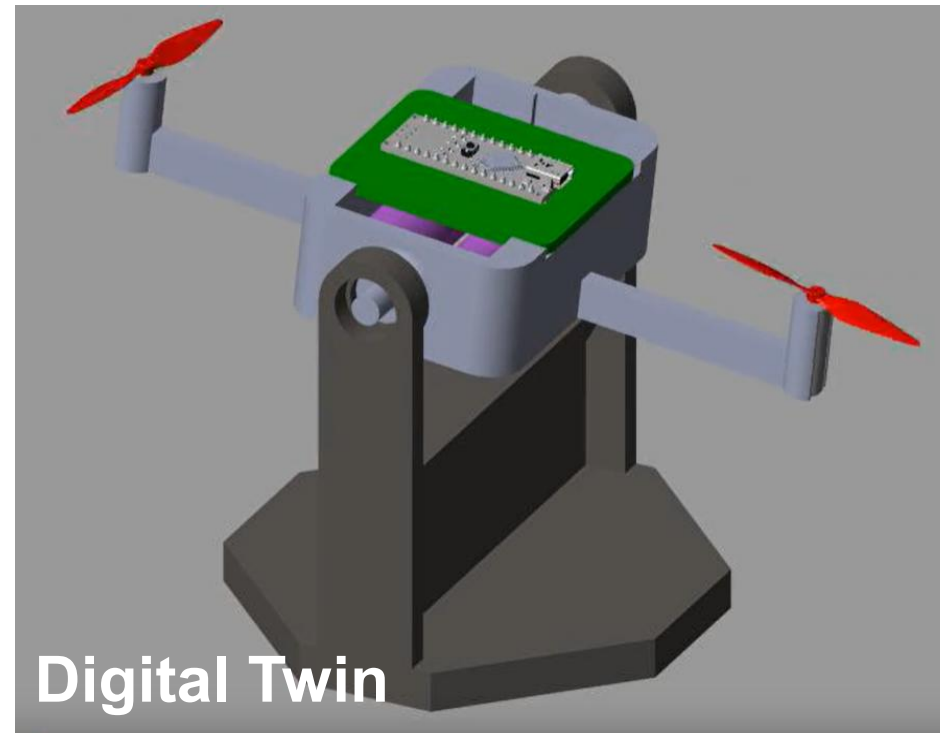
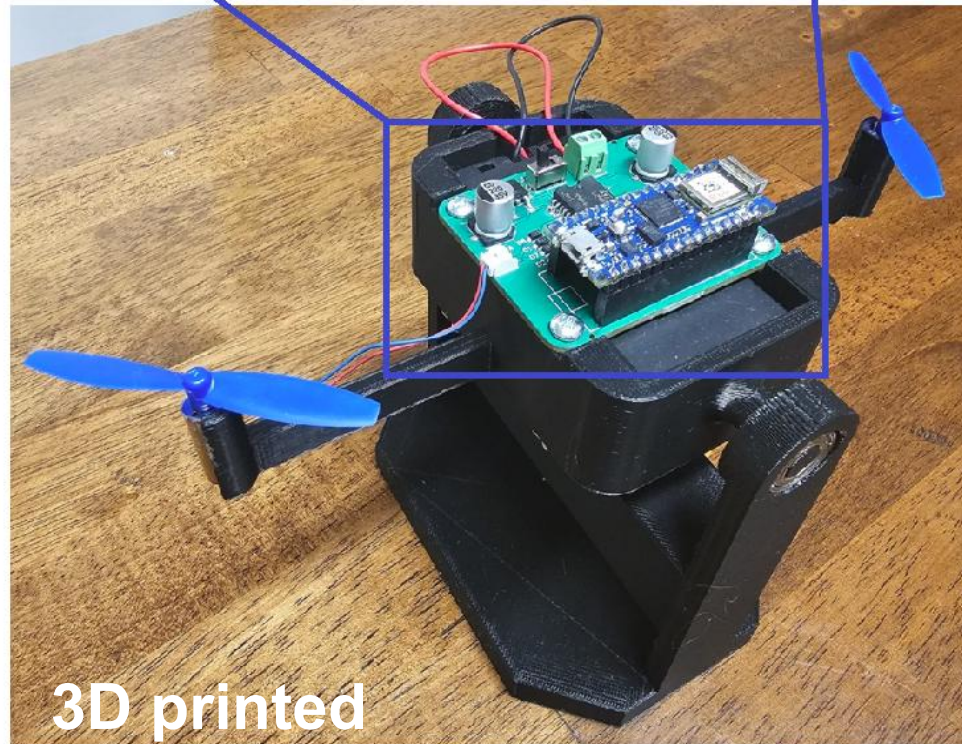
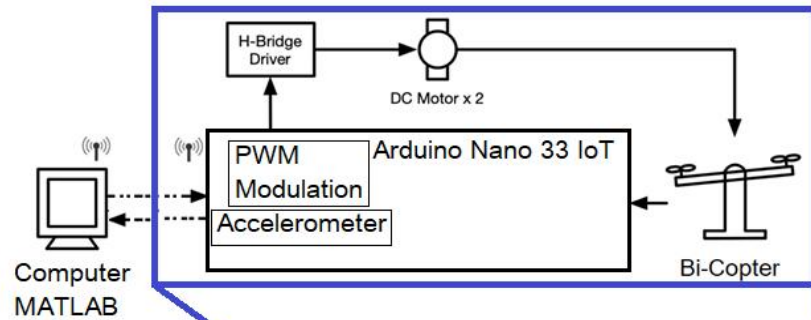
**Modern (MPC)
Controller
Design**

Hardware overview

Wireless
operation



The low-cost bi-copter and its features

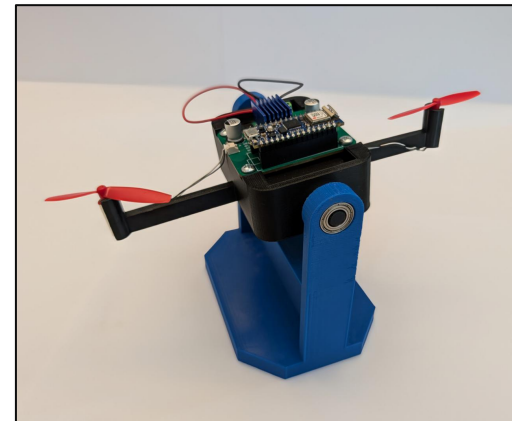
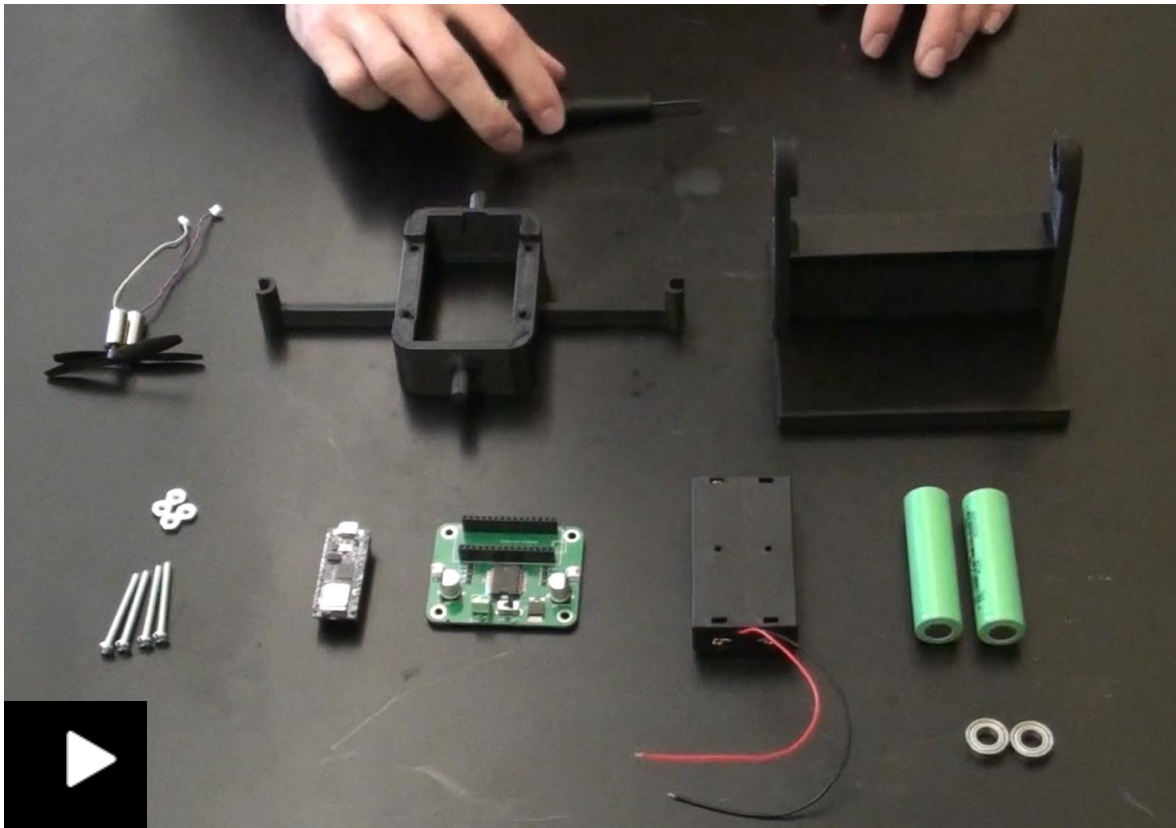


3D printing and assembly instructions are available for the bi-copter hardware

Assembly instructions [\[link\]](#)

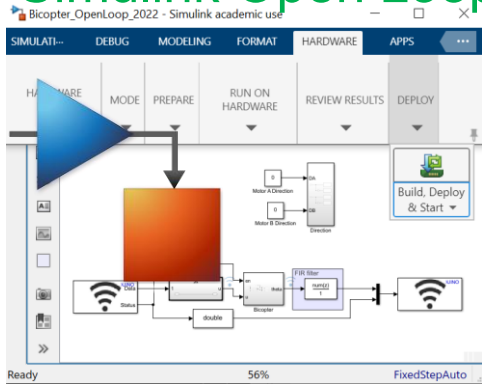
3D printing instructions [\[link\]](#)

Source Files [\[link\]](#)

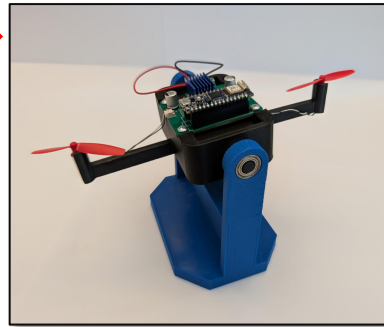
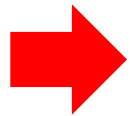


Bi-copter runs a pre-loaded Simulink controller (open- or closed-loop) and sends/receives data to MATLAB Live Script

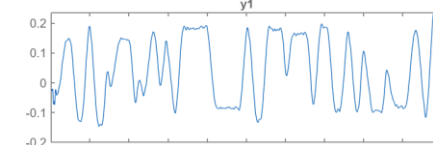
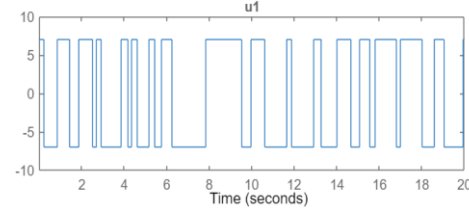
Simulink Open Loop



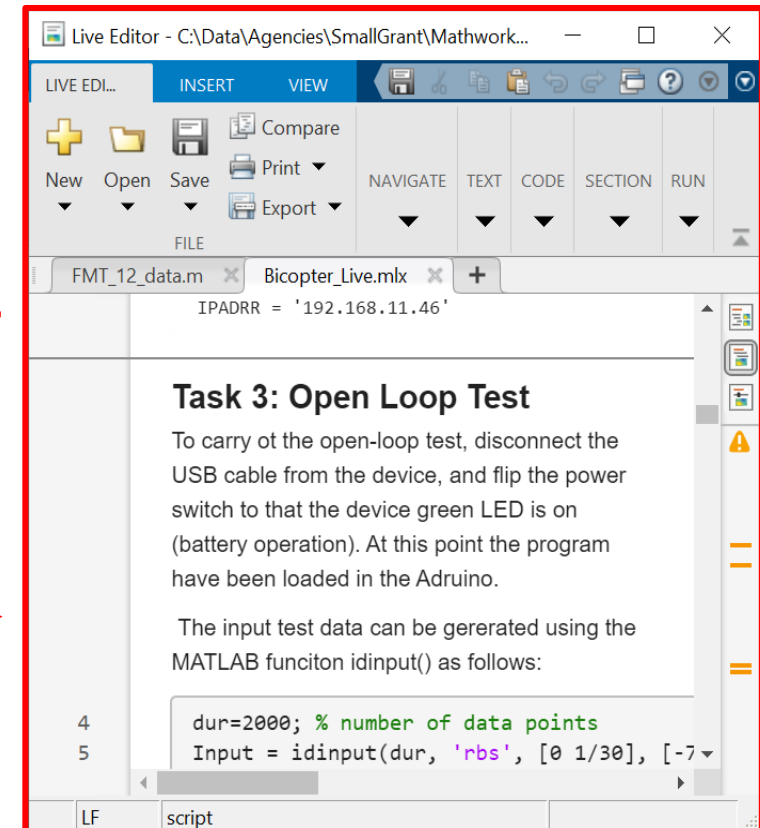
Step 1: Deploy Controller



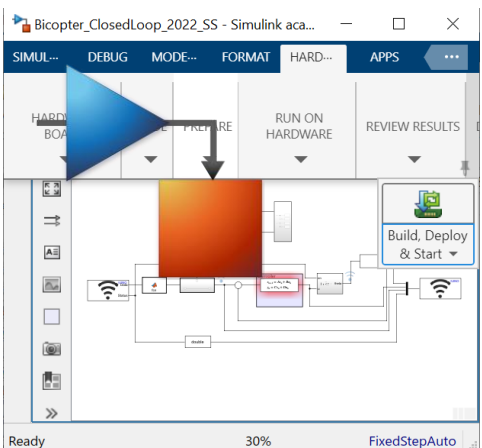
Step 2: Run Live Script



Live Script



Simulink Closed-Loop



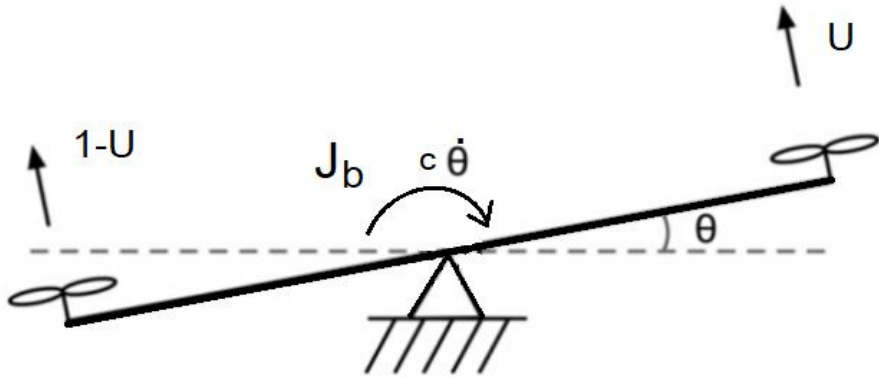
Parameter Updates Via Workspace

Workflow for classical controller design

Workflow



Modeling based on physics (linear approximation)



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

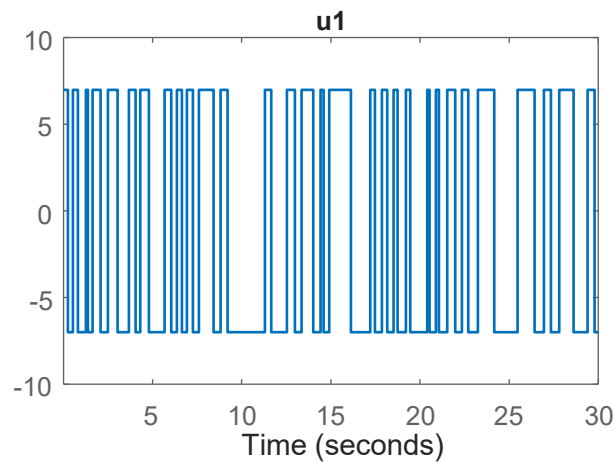
$$\text{DC Motors: } L \frac{di}{dt} + Ri + K_b \omega_m = v, \quad (2)$$

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

Overall Model:

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

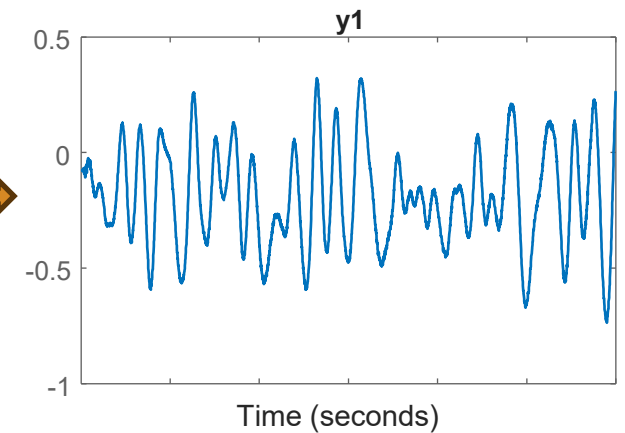
Design of experiments are performed to collect input-output data for system identification



Excite



Collect response



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



Live Editor - C:\Data\Agencies\SmallGrant\Mathworks\BiCopter\Bicopter_Live.mlx

LIVE EDITOR INSERT FIGURE VIEW

2 `close all`
3 `IPADRR='192.168.11.46'`

`IPADRR = '192.168.11.46'`

Task 3: Open Loop Test

To carry out the open-loop test, disconnect the USB cable from the device, and flip the power switch to that the device green LED is on (battery have been loaded in the Arduino).

The input test data can be generated using the MATLAB function `idinput()` as follows:

```
4 dur=2000; % number of data points
5 Input = idinput(dur, 'rbs', [0 1/30], [-7 7]);
```

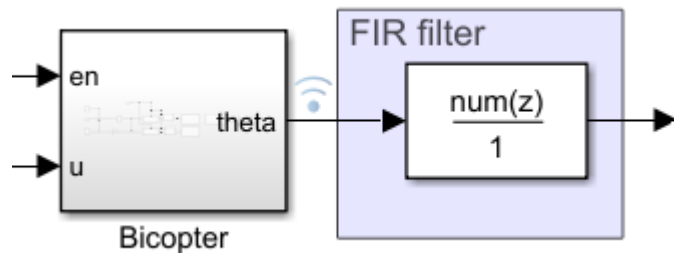
This generates a binary input with a frequency bandwidth from 0 to 0.1 of the Nyquist frequency and a range from -10 to 10. Once data has been generated, run the following script to send it to the Arduino and collect response. **Make sure to keep clear from the propellers before running the following sequence!!!**

```
6 tcp = tcpclient(IPADRR,25000);
7 %write(tcp,0,"int8");
8 %pause(1);
9 flush(tcp);
10 write(tcp,Input,'int8');
11 tic
12 en = 0;
13 while en == 0
14     dat = read(tcp,2,'double');
15     en = dat(2);
16 end
17 D = read(tcp,2*dur,'double');
18 Output = D(1:2:end);
```

|| emailarizona-my.sharepoint.com is sharing your screen and audio. [Stop sharing](#) [Hide](#)

Zoom: 100% UTF-8 LF script Ln 7 Col 22

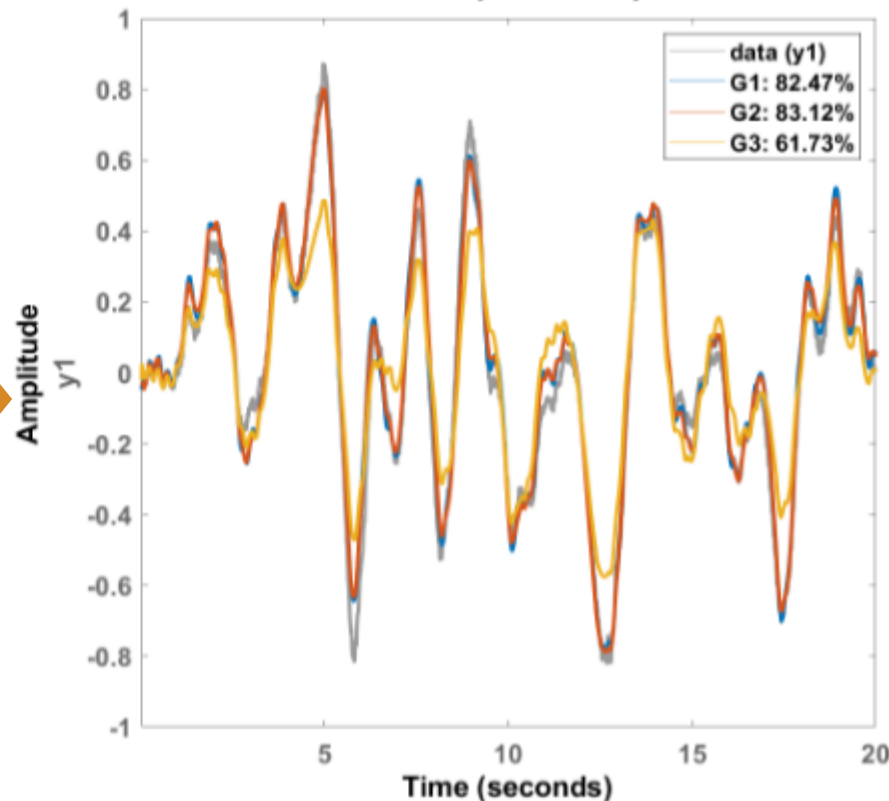
Select most appropriate model



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)
```

Simulated Response Comparison



PID controller is tuned using different gain cross-over frequencies

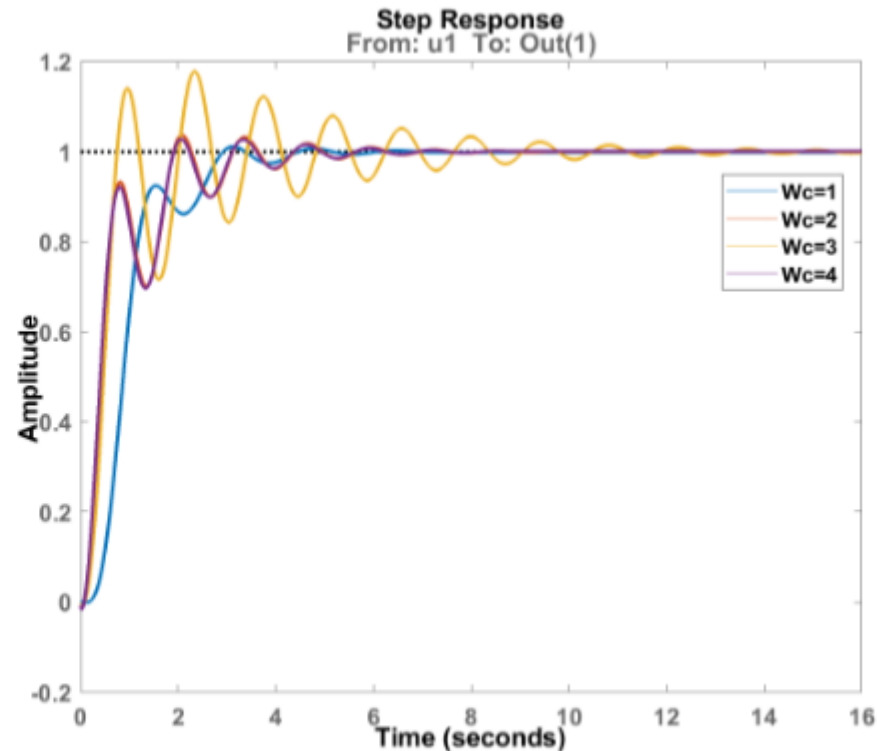


PIDtune

```
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))
```

$$C2 = K_p + K_i \cdot \frac{T_s \cdot z}{z-1} + K_d \cdot \frac{z-1}{T_s \cdot z}$$

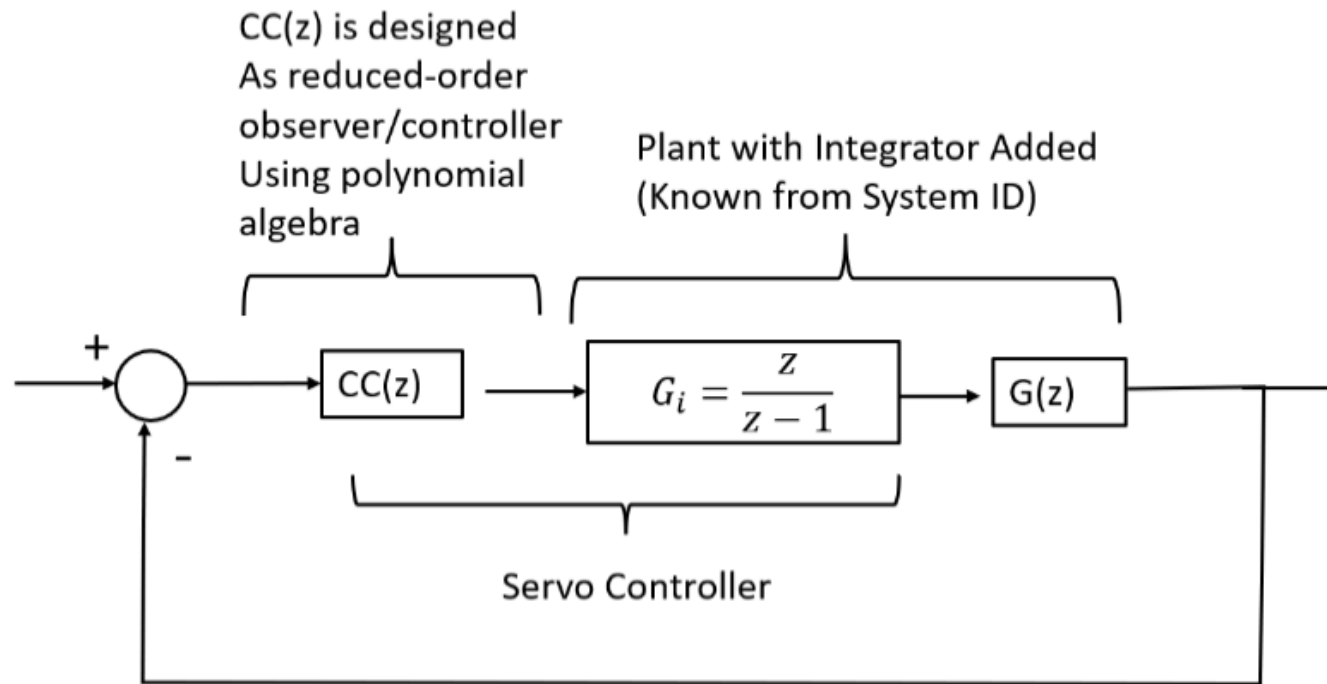
with $K_p = -6.79$, $K_i = -23.1$, $K_d = -0.499$, $T_s = 0.01$



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!



Live Editor - C:\Data\Agencies\SmallGrant\Mathworks\BiCopter\BiCopter_Live.mlx

Task 5: Controller Design and Implementation

After fitting a model to the bicopter i/o data, we proceed with the design of a controller. Use Tune PID Controller Live Task (or iin command module below)

```
31 C0 = pid(1,1,1,'Ts',ts,'Formula','BackwardEuler','Formula','BackwardEuler');
32 C=pidtune(G,C0,1.5)
```

C =

$$K_p + K_i * \frac{T_s z}{z-1} + K_d * \frac{z-1}{T_s z}$$

with $K_p = -5.82$, $K_i = -71.5$, $K_d = -0.118$, $T_s = 0.01$

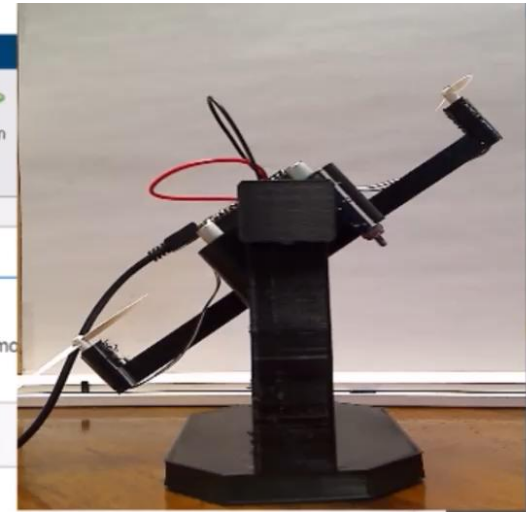
Sample time: 0.01 seconds
Discrete-time PID controller in parallel form.

```
33 figure;step(feedback(C*G,.5))
```

Step Response
From: u1 To: Out(1)

emilarizona-my.sharepoint.com is sharing your screen and audio. [Stop sharing](#) [Hide](#)

Zoom: 100% UTF-8 LF script Ln 28 Col 20



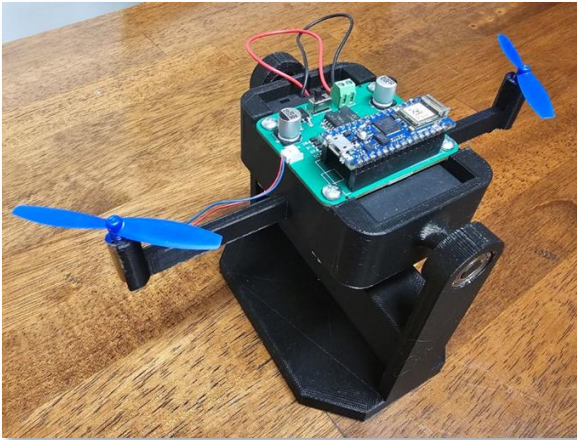
Closed-loop testing of the servo controller



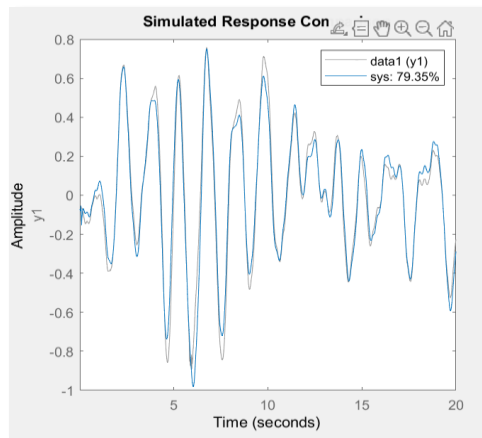
The screenshot displays the MATLAB/Simulink environment for a Bicopter model. The top toolbar includes tabs for SIMULATION, DEBUG, MODELING, FORMAT, HARDWARE, and APPS. The HARDWARE tab is active, showing the Hardware Board set to 'Arduino Nano 33 IoT' and options to 'Run on board', 'Hardware Settings', 'Log Signals', and 'Add Viewer'. The main workspace shows the Simulink model 'Bicopter_ClosedLoop_2022_SS'. The model includes a 'Read Setpoint' block, a 'Fun' block, a 'Controller' block with the equations $x_{k+1} = Ax_k + Bu_k$ and $y_k = Cx_k + Du_k$, and a 'Beta' block. The 'Controller' block is highlighted with a blue border. The 'Diagnostic Viewer' at the bottom shows a 'Build process completed successfully' message and a 'Build Summary' table. A sharing notification from emailarizona-my.sharepoint.com is also visible.

Model	Action	Rebuild Reason
Bicopter_ClosedLoop_2022_SS	Code generated and c	

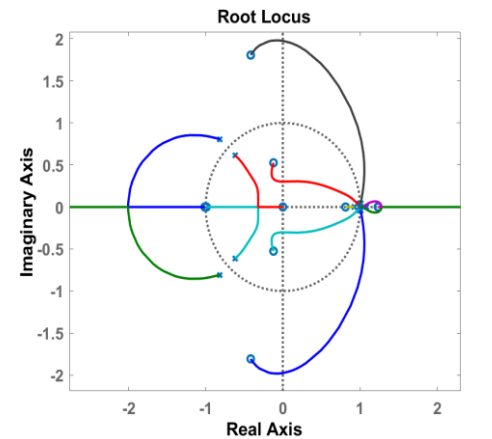
Next, we will focus on MPC design and deployment



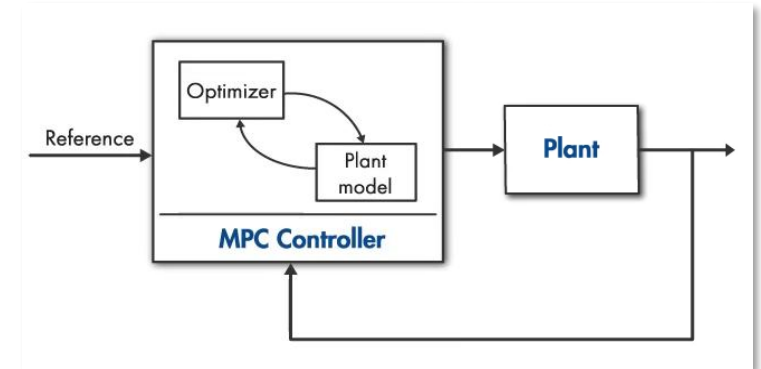
Bi-Copter



**Modeling and
System
Identification**

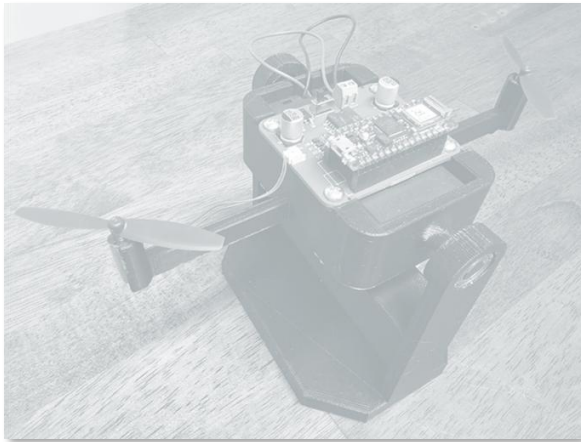


**Classical
Controller
Design**

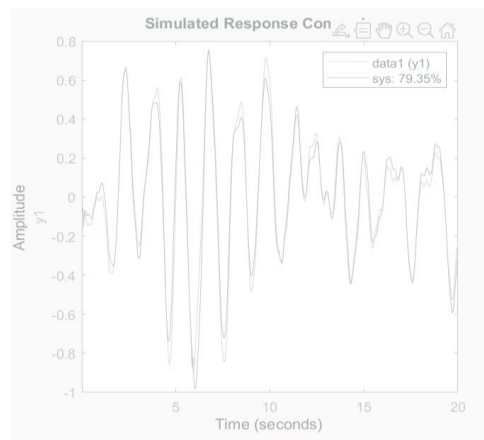


**Modern (MPC)
Controller
Design**

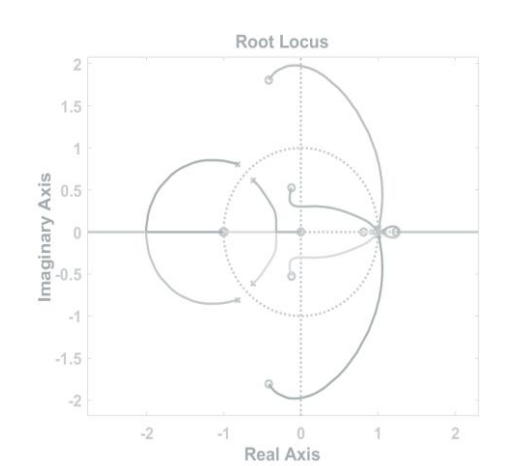
Next, we will focus on MPC design and deployment



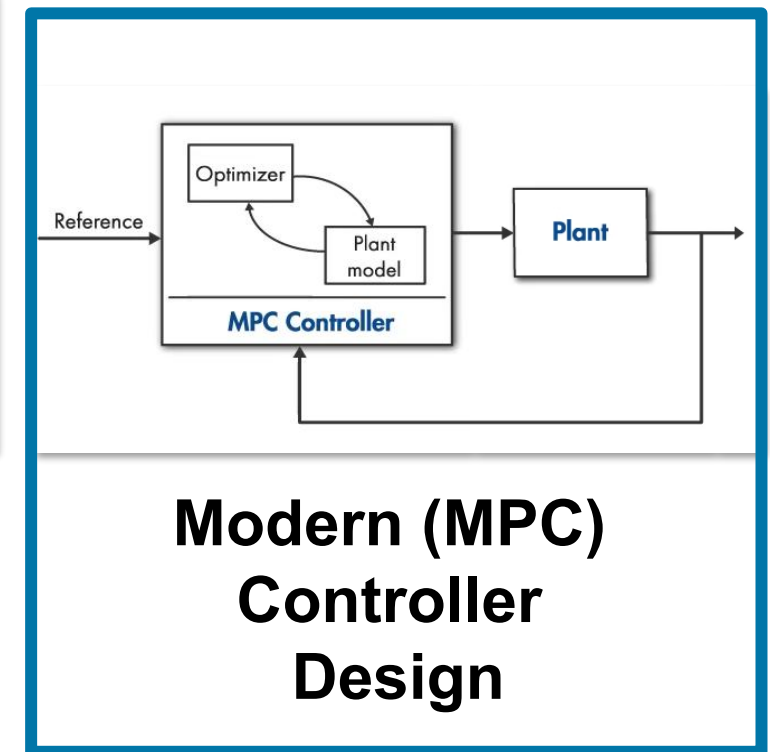
Bi-Copter



Modeling and
System
Identification

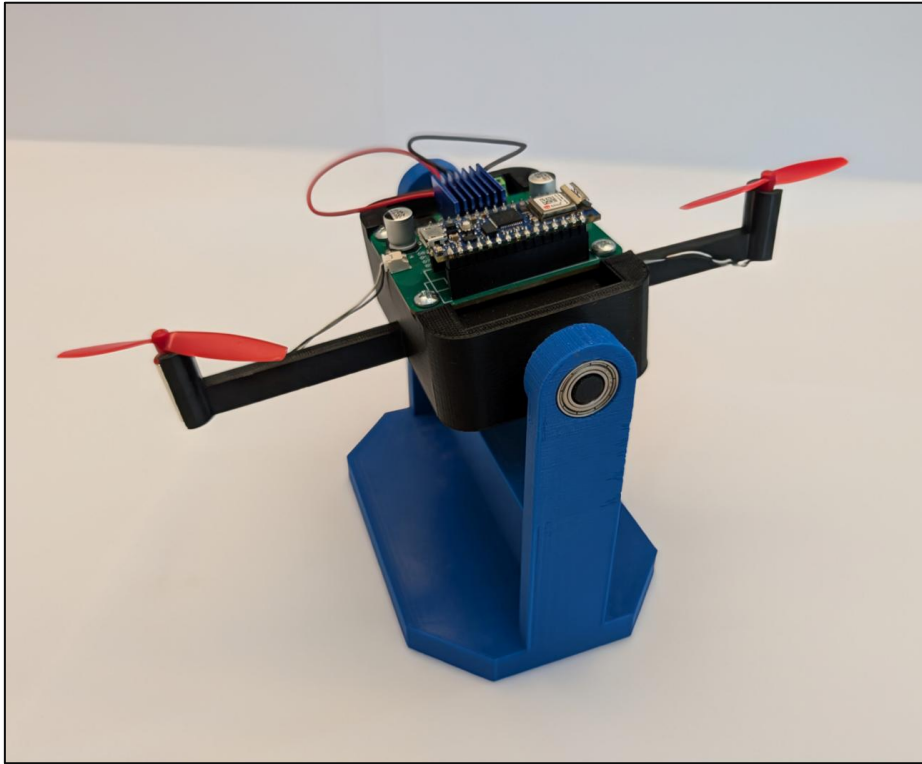


Classical
Controller
Design



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

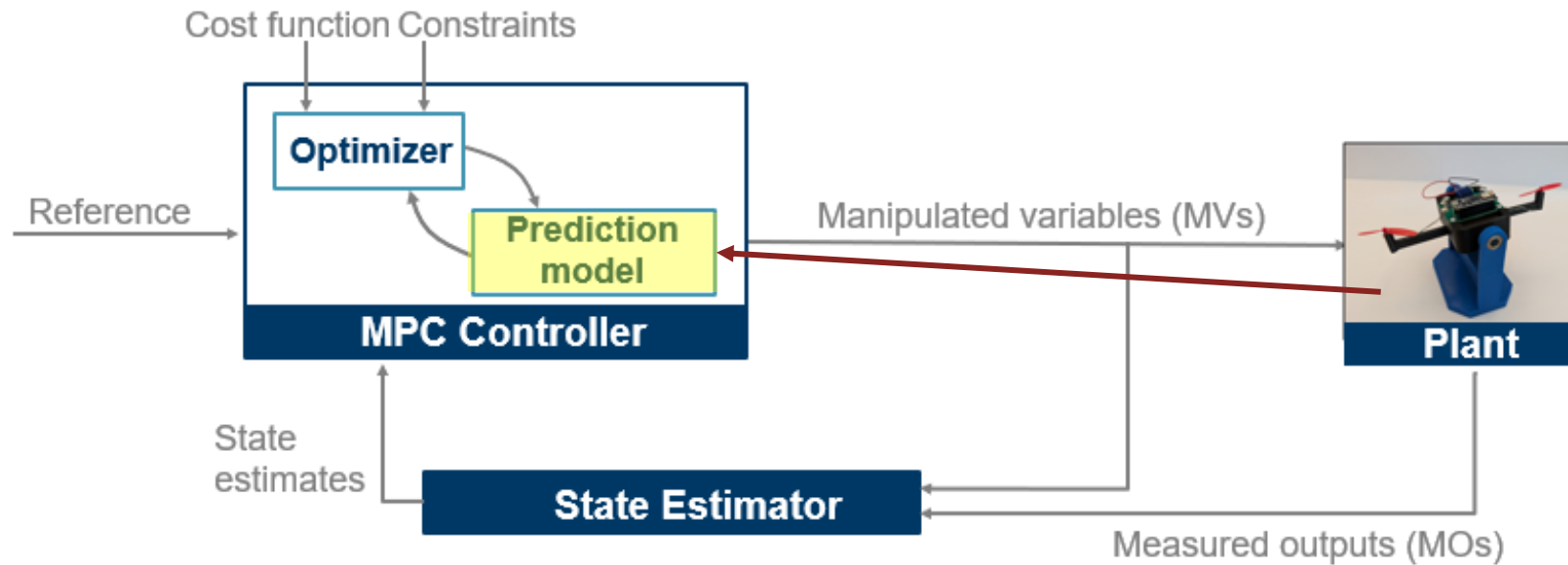
Model predictive control (MPC)

Mechatronics

Hands-on learning

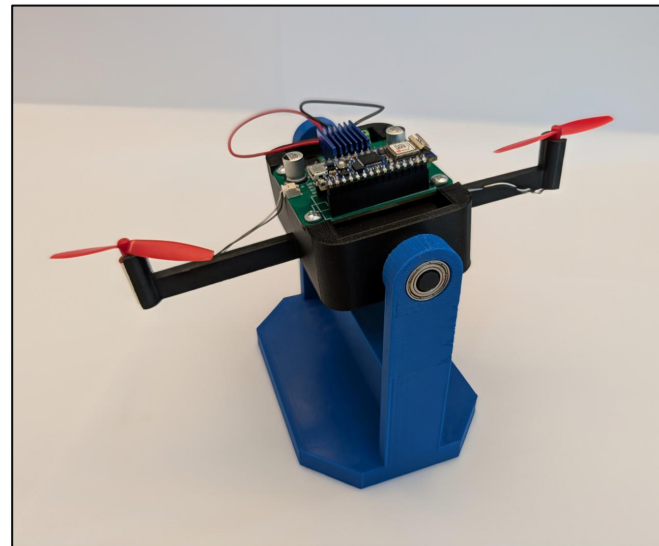
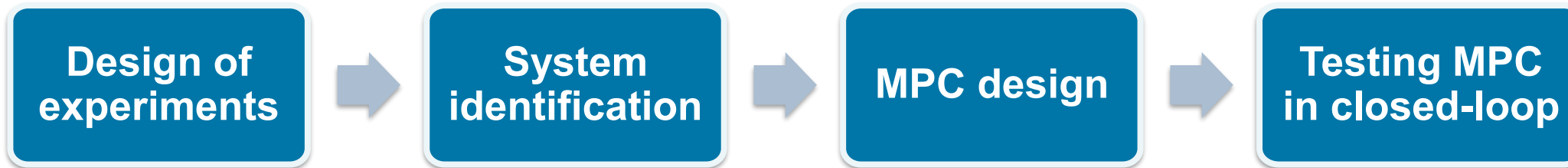
Sensors and actuators

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



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

Workflow

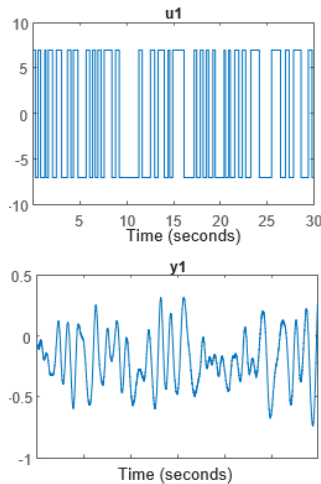
Design of experiments

System identification

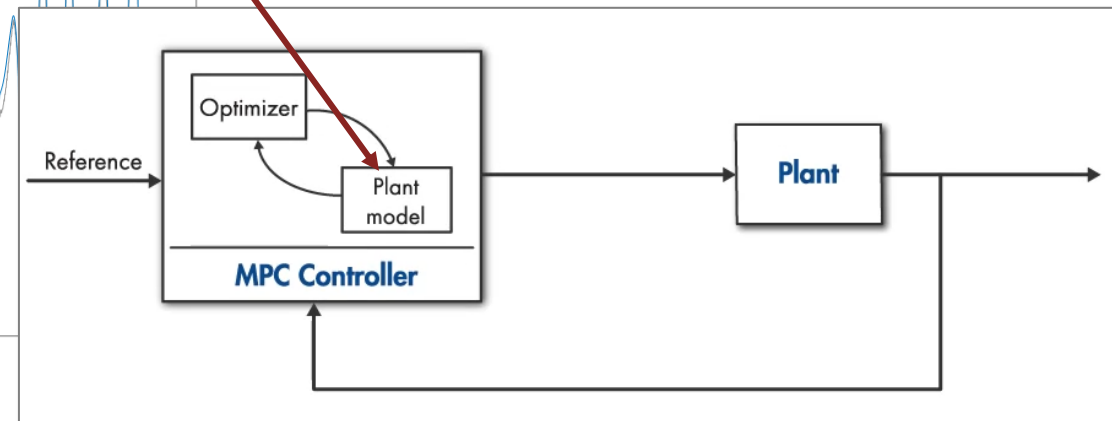
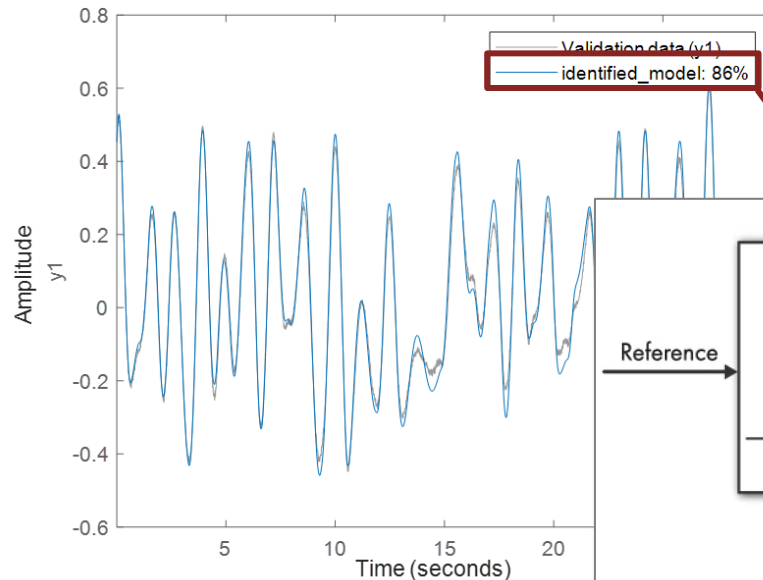
MPC design

Testing MPC in closed-loop

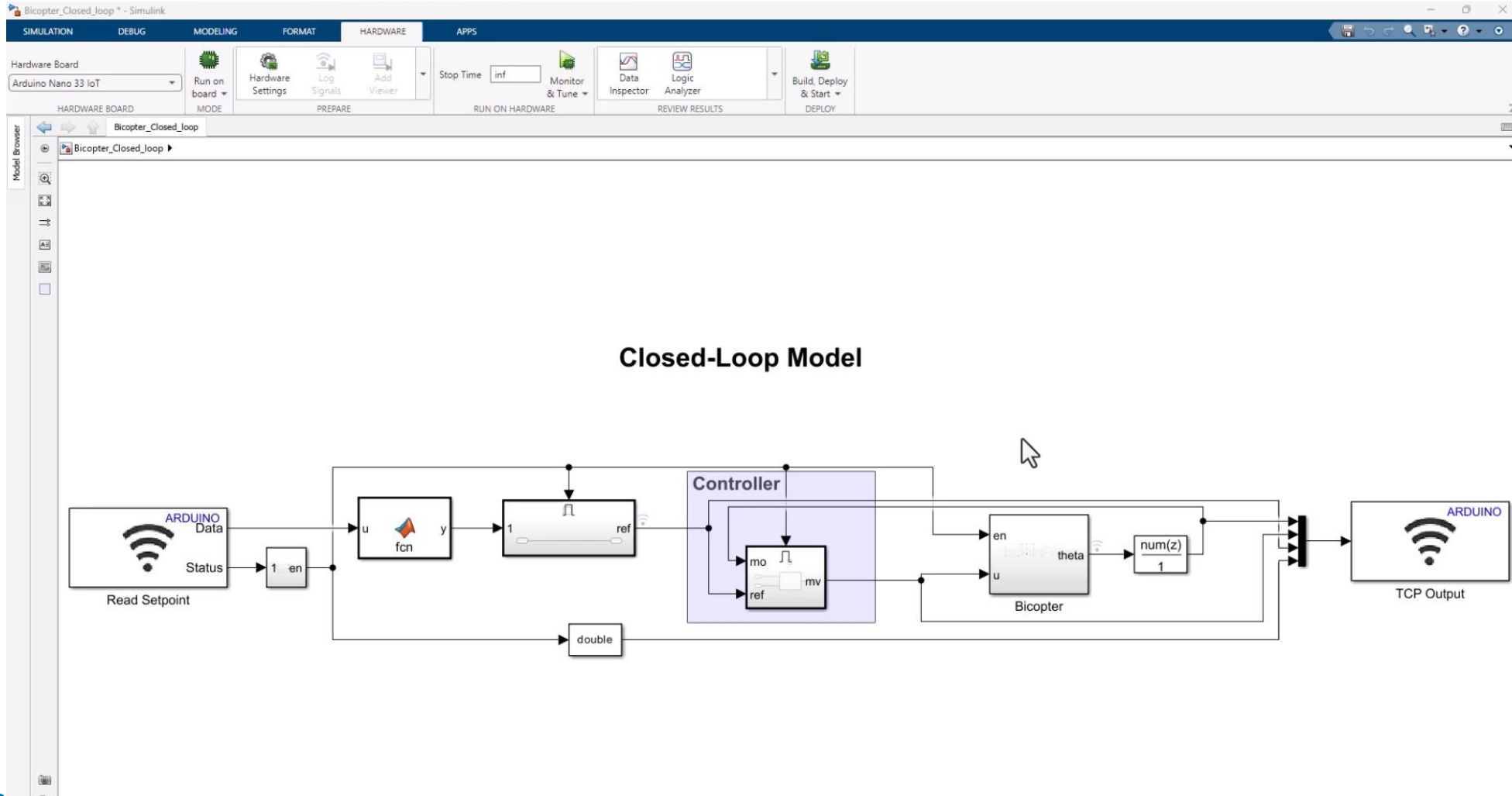
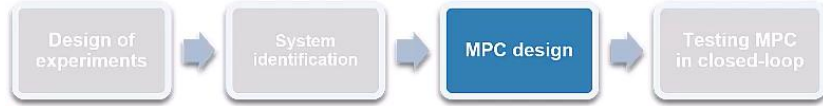
Input-output data



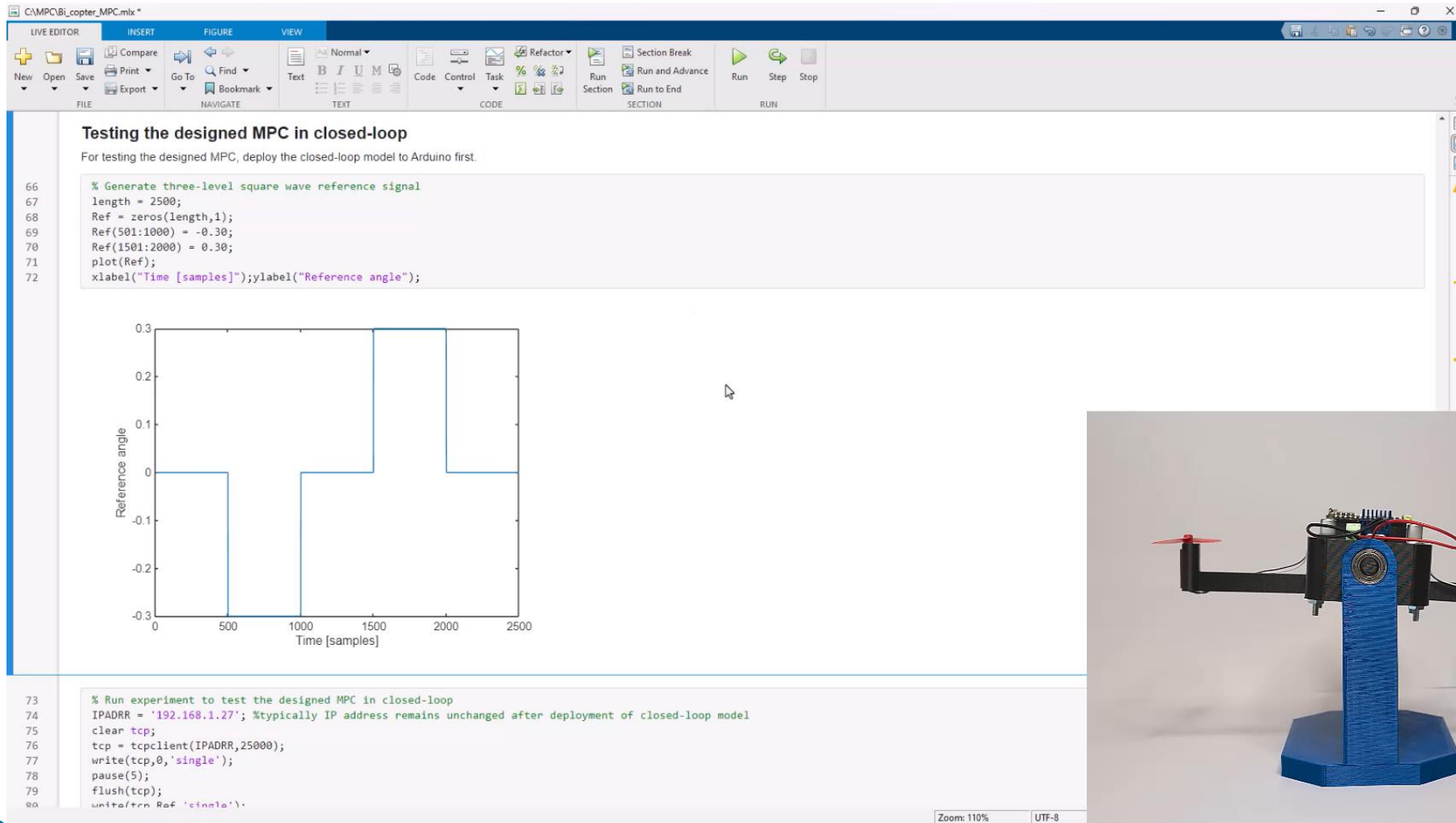
Simulated Response Comparison



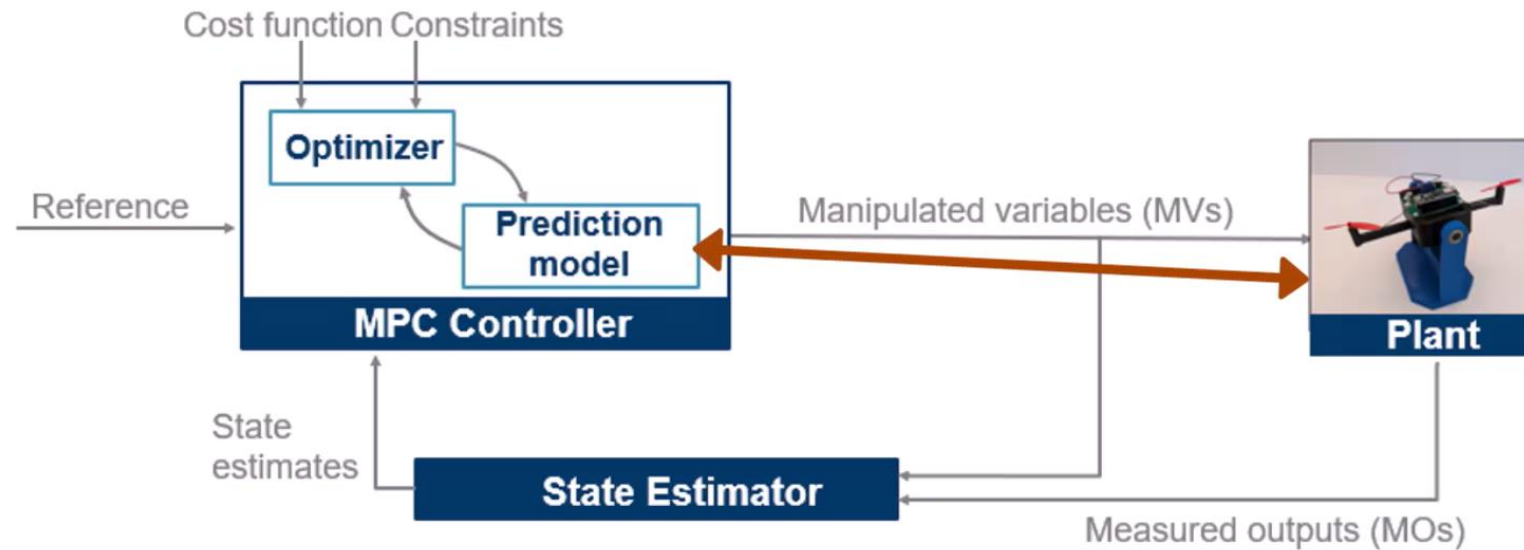
An MPC is designed using the MPC Designer app to maintain the bi-copter at the specified reference angles.



Testing the deployed MPC in closed-loop



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



MathWorks ✓

@MathWorks

Share the EXPO experience
#MATLABEXPO



in/eniko-enikov-
618b8b106



in/melda-ulusoy

MATLAB EXPO

Thank you



© 2025 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

