

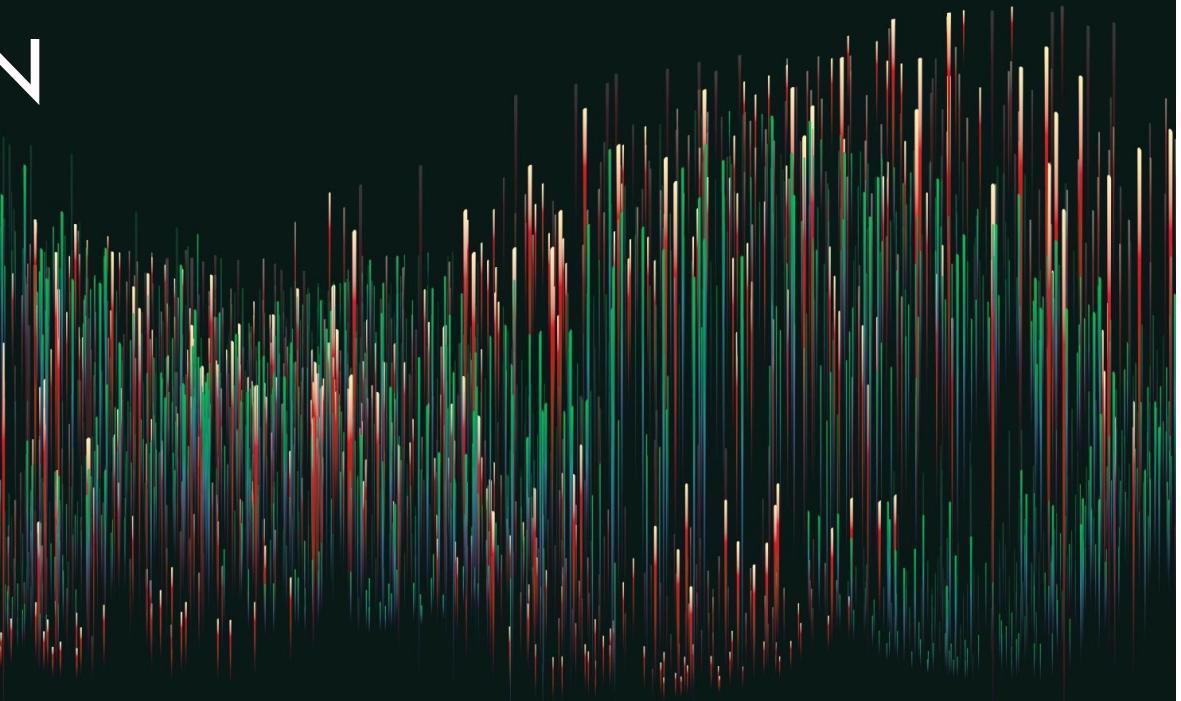
สำหรับการฝึกอบรมเชิงปฏิบัติการ "อุปกรณ์ไอโอที่สำหรับงานควบคุมอุตสาหกรรม"
ภาควิชาพิสิกส์ คณะวิทยาศาสตร์ ม.นเรศวร 26-27 มิถุนายน 2564

การพัฒนาตัวควบคุม PID โดย MICROPYTHON บน ESP32

DR. VARODOM TOOCHINDA

DEPT. OF MECHANICAL
ENGINEERING

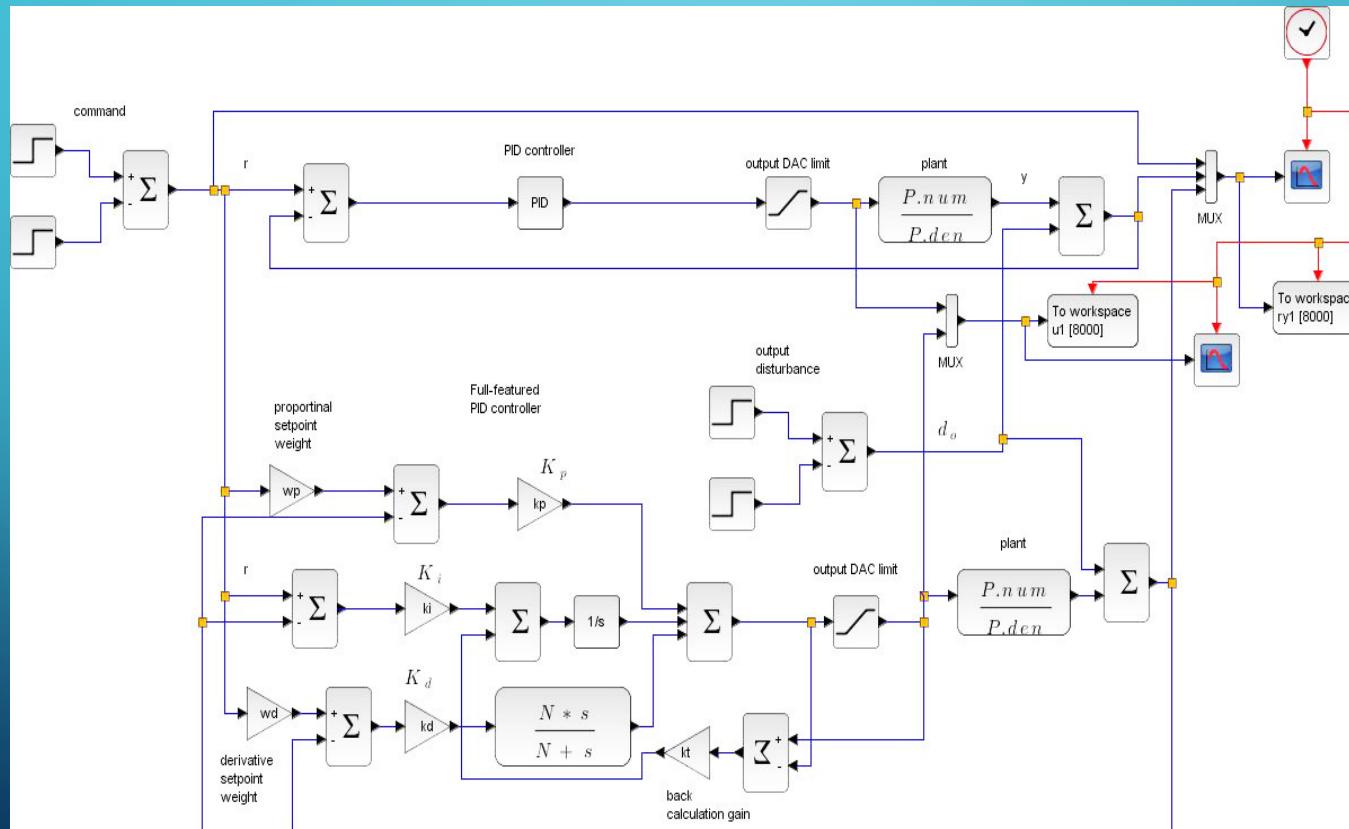
KASETSART UNIVERSITY



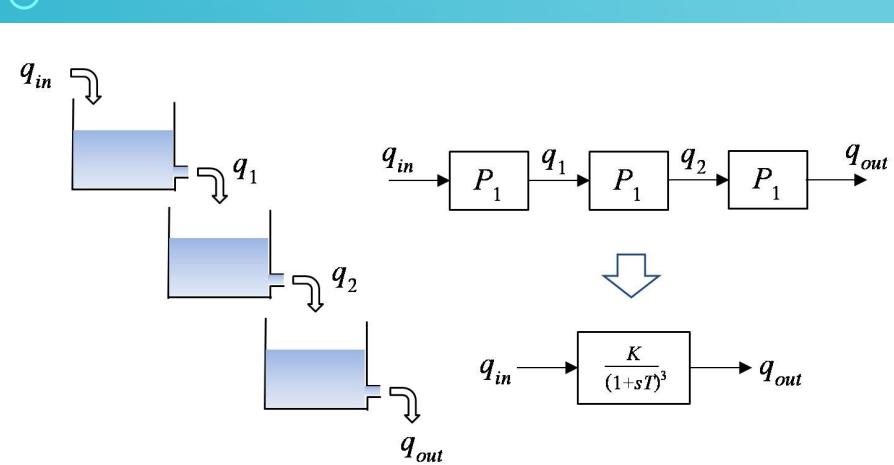
หัวข้อ

- พัฒนาตัวควบคุม PID พร้อมฟังก์ชันเสริม โดย micropython
- สร้าง Freeboard สำหรับควบคุม
- ทดสอบผลตอบสนองขั้นบันได เก็บข้อมูลเปรียบเทียบบน Jupyter notebook
- ปรับค่าพารามิเตอร์ควบคุม และดูผลตอบสนอง

โครงสร้างของตัวควบคุม PID และฟังก์ชันเสริม

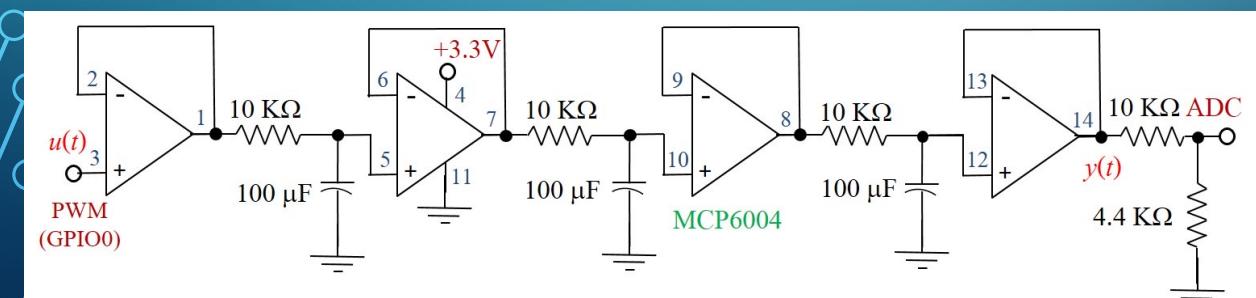
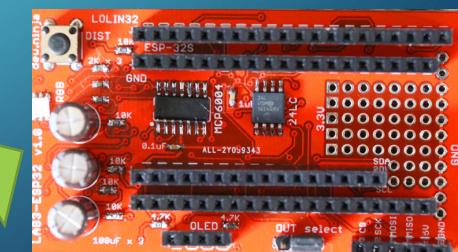


ทดสอบกับบอร์ด LAG3-ESP32 (หรือจำลองผ่านต์โดยอัลกอริทึม)



normalized transfer
function

$$\frac{1}{(s+1)^3}$$



แปลงตัวควบคุมในระบบต่อเนื่องเป็นดิจิตอล (1)

$$u = K_p e + K_i e + K_t lus + \frac{K_d N s}{s+N} e_d(s)$$

$$= u_{p0} + \frac{s}{s+N} u_{i0} + u_{do}$$

Anti Windup

Bilinear Transform

$$s = \frac{2(z-1)}{T(z+1)}$$

$$u_{p0} = K_p e_p ; \quad e_p = w_p r - y \quad (1)$$

$$u_{i0} = \frac{K_i T(z+1)}{2(z-1)} e + \frac{K_t T(z+1)}{2(z-1)} lus ; \quad lus = u_{SAT} - u$$

$$u_{i1} = u_{i0}(z) \quad = \frac{0.5 T(1+z^{-1})}{1-z^{-1}} e + \frac{0.5 T(1+z^{-1})}{1-z^{-1}} lus = \frac{0.5 T(1+z^{-1})(K_i e + K_t lus)}{1-z^{-1}} +$$

$$u_{i0} - u_{i1} = \underbrace{0.5 T K_i}_{\triangleq b_i} l_0 + \underbrace{0.5 T K_i}_{\triangleq b_i} e_1 + \underbrace{0.5 T K_t lus_0}_{\triangleq b_t} + \underbrace{0.5 T K_t lus_1}_{\triangleq b_t}$$

$$u_{i0} = u_{i1} + b_i(l_0 + e_1) + b_t(lus_0 + lus_1) \quad (2)$$

$$u_d = \frac{K_d N (1-z^{-1})}{(1+0.5 NT) + (0.5 NT-1) z^{-1}} e_d ; \quad e_d = w_d r - y$$

$$(ad_1 + ad_2 z^{-1}) u_d = K_d N e_d - K_d N z^{-1} \rightarrow ad_1 u_{d0} + ad_2 u_{d1} = K_d N e_{d0} - K_d N e_{d1}$$

$$u_{d0} = - \frac{ad_2 u_{d1}}{ad_1} + \frac{K_d N e_{d0} - K_d N e_{d1}}{ad_1} \quad u_{d0} = ad_1 u_{d1} + b_d (e_{d0} - e_{d1}) \quad (3)$$

แปลงตัวควบคุมในระบบต่อเนื่องเป็นดิจิตอล (2)

$$U_o = U_{p0} + U_{i0} + U_{d0}$$

$$U_{p0} = K_p l_p \quad ; \quad l_p = W_p \gamma - \gamma$$

$$U_{i0} = U_{i1} + b_i (l_0 + l_1) + b_t (l_{us0} + l_{us1}) \quad ; \quad l_{us} = U_{sat} - U$$

$$U_{d0} = a_d U_{d1} + b_d (l_{d0} - l_{d1}) \quad ; \quad l_d = W_d \gamma - \gamma$$

$$b_i = 0.5 T k_i$$

$$b_t = 0.5 T K_T$$

$$a_{d1} = (1 + 0.5 NT) \quad b_d = \frac{K_d N}{a_{d1}}$$

$$a_{d2} = (0.5 NT - 1)$$

$$a_d = -\frac{a_{d2}}{a_{d1}}$$

ฟังก์ชันคำนวณสัมประสิทธิ์

```
def PID_update():
    global ad, bd, bi, bt
    bi = 0.5*T*ki
    bt = 0.5*T*kt
    ad1 = 1+0.5*N*T
    ad2 = 0.5*N*T - 1
    ad = -ad2/ad1
    bd = kd*N/ad1
    update_freeboard()
```

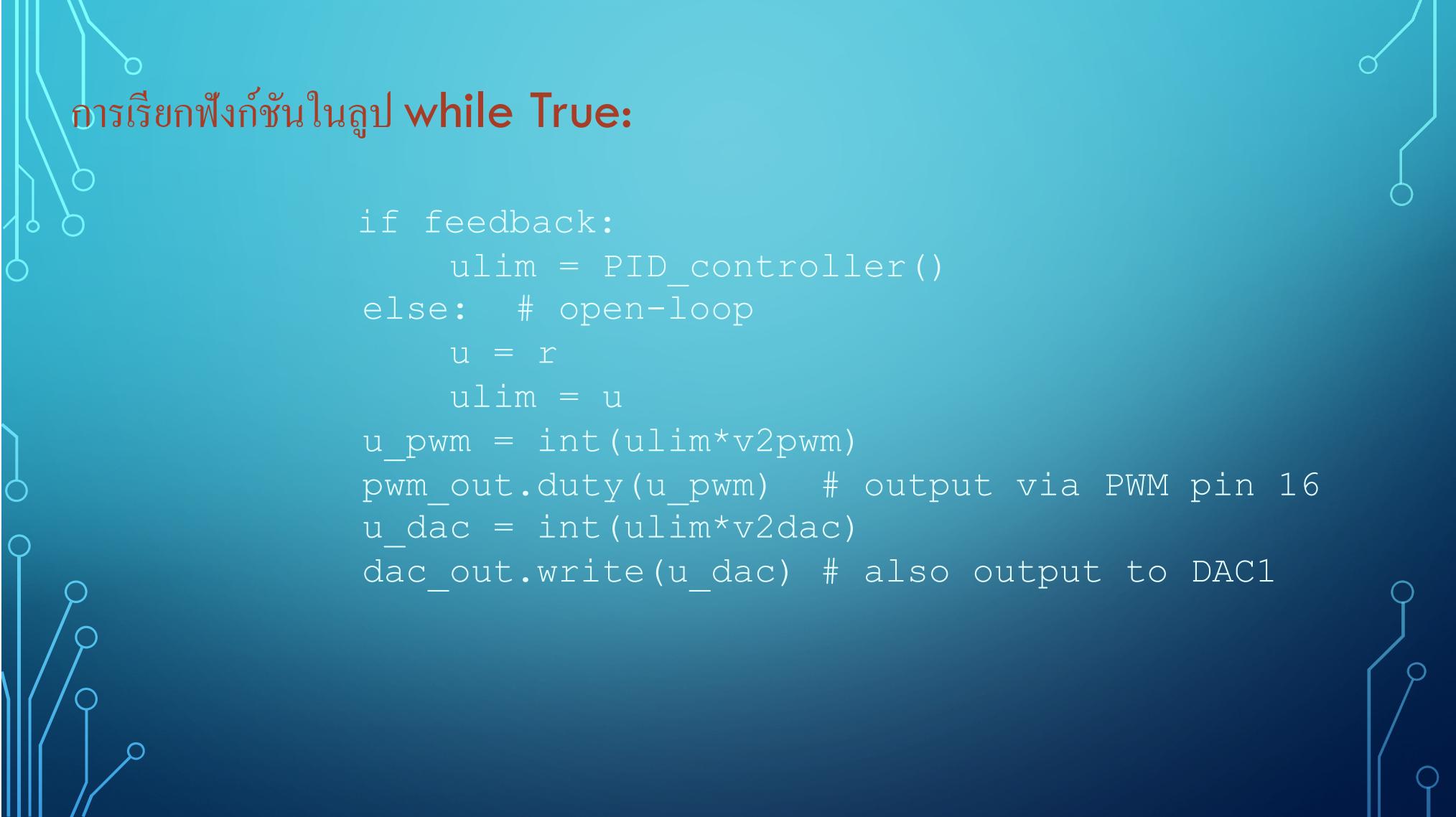
ฟังก์ชันตัวควบคุม PID

```
def PID_controller():
    global e1,e0,ed1,ed0, eus1,eus0,
    uil, ui0, ud1, ud0, u

    # state transfer
    e1 = e0
    ed1 = ed0
    eus1 = eus0

    ui1 = ui0
    ud1 = ud0
    # compute errors for each term
    e0 = r - y
    ep0 = wp*r - y # weighted
                    # proportional error
    ed0 = wd*r - y # weighted
                    # derivative error
```

```
    up0 = kp*ep0 # output of P term
    ui0 = ui1 +bi*(e0+e1) + bt*(eus0+eus1)
    # output of I term
    ud0 = ad*ud1 +bd*(ed0 - ed1) # output
    of D term
    u = up0 + ui0 + ud0
    u_lim = u
    if u > UMID:
        eus0 = UMID - u # compute error
                           # for back calculation term
        u_lim = UMID           # limit u to
                               # UMID
    elif u < -UMID:
        eus0 = -u - UMID # compute error
                           # for back calculation term
        u_lim = -UMID           # limit u to
                               #-UMID
    u_lim += UMID
    return u_lim
```



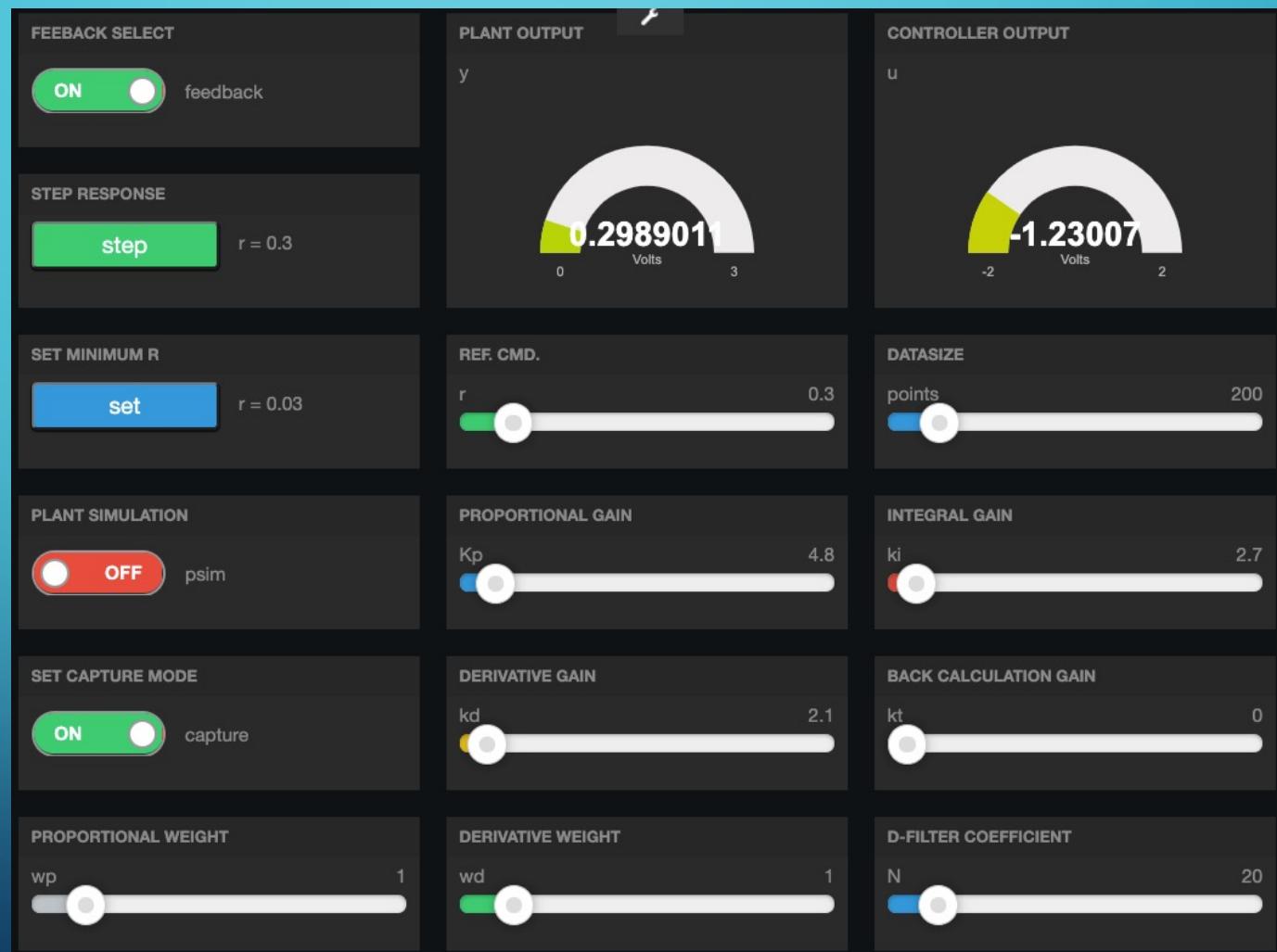
การเรียกฟังก์ชันในลูป while True:

```
if feedback:  
    ulim = PID_controller()  
else: # open-loop  
    u = r  
    ulim = u  
u_pwm = int(ulim*v2pwm)  
pwm_out.duty(u_pwm) # output via PWM pin 16  
u_dac = int(ulim*v2dac)  
dac_out.write(u_dac) # also output to DAC1
```

เพิ่มเงื่อนไขรับคำสั่งใน cmdInt()

```
:  
elif cmdstr.lower() == "kp":  
    if noparm==1:  
        print ("Current kp =  
{ }".format(kp))  
    else:  
        kp = float(parmstr)  
        if kp > 100: # set maximum kp  
            kp = 100  
        elif kp < 0: # minimum kp  
            kp = 0  
        PID_update()  
  
elif cmdstr.lower() == "ki":  
:
```

Freeboard for PID



ทดสอบผลตอบสนองขั้นบันได

The screenshot shows a web-based control interface for a system. On the left, there is a sidebar with various tabs and links. The main area contains several control panels:

- FEEDBACK SELECT:** A switch labeled "ON feedback".
- STEP RESPONSE:** A button labeled "step" with a value "r = 0.3".
- SET MINIMUM R:** A button labeled "set" with a value "r = 0.03".
- PLANT SIMULATION:** A switch labeled "OFF psim".
- SET CAPTURE MODE:** A switch labeled "ON capture".
- PROPORTIONAL WEIGHT:** A slider labeled "wp" with a value of 1.

On the right side, there are two large circular displays:

- PLANT OUTPUT:** Shows a value of "0.0290109" with a scale from 0 to 3 Volts.
- CONTROLLER OUTPUT:** Shows a value of "1.51240" with a scale from -2 to 2 Volts.

Below these displays are several gain sliders:

- REF. CMD.:** A slider labeled "r" with a value of 0.
- DATASIZE:** A slider labeled "points" with a value of 200.
- PROPORTIONAL GAIN:** A slider labeled "Kp" with a value of 4.8.
- DERIVATIVE GAIN:** A slider labeled "kd" with a value of 2.1.
- INTEGRAL GAIN:** A slider labeled "ki" with a value of 2.7.
- BACK CALCULATION GAIN:** A slider labeled "kt" with a value of 0.
- PROPORTIONAL WEIGHT:** A slider labeled "wp" with a value of 1.
- DERIVATIVE WEIGHT:** A slider labeled "wd" with a value of 0.

At the bottom center, there is a small image of an ESP32 development board.

On the far right, there is a code editor window titled "g3_pid_netpie.py" containing Python code for a WiFi connection and plant simulation. A video player window shows a man sitting at a desk with a guitar.

```
t_current = 0
t_previous = 0
dt = 0
# variables for plant simulation
a = 2+T
b = T-2
y_states = [0.0]*6
u_states = [0.0]*6
#####
cmdtime_current = 0 # This delay is needed for node red
cmdtime_prev = 0
CMD_DELAY = 1000

def wifi_connect():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('connecting to network...')
        wlan.connect(wifi_ssid, wifi_pwd)
        while not wlan.isconnected():
            pass
    print('network config:', wlan.ifconfig())
    shell >
datamat = np.array([
[0,0.3,0.02901099,7.330201,3.3],
[0.05,0.3,0.02901099,2.318924,3.3],
[0.1,0.3,0.02813187,0.7021441,2.352144],
[0.15,0.3,0.03164835,0.04646528,1.696465],
[0.2,0.3,0.03164835,-0.04357779,1.606422],
[0.25,0.3,0.03340659,-0.1068736,1.543126],
[0.3,0.3,0.03868132,-0.2249923,1.425008],
[0.35,0.3,0.04043956,-0.1422761,1.507724],
[0.4,0.3,0.04483516,-0.1834713,1.466529],
[0.45,0.3,0.05274725,-0.3039473,1.346053],
[0.5,0.3,0.05714285,-0.2319831,1.418017],
[0.55,0.3,0.06505495,-0.3158633,1.334137],
[0.6,0.3,0.07208791,-0.3195103,1.33049],
[0.65,0.3,0.08087912,-0.3808489,1.269151],
```

คัดลอกใส่ในเซลของ Jupyter notebook

Jupyter plot_pid_stepcompare Last Checkpoint: 16 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.7 (controlenv)

In [34]:

```
import numpy as np
import matplotlib.pyplot as plt
import control as ctl
```

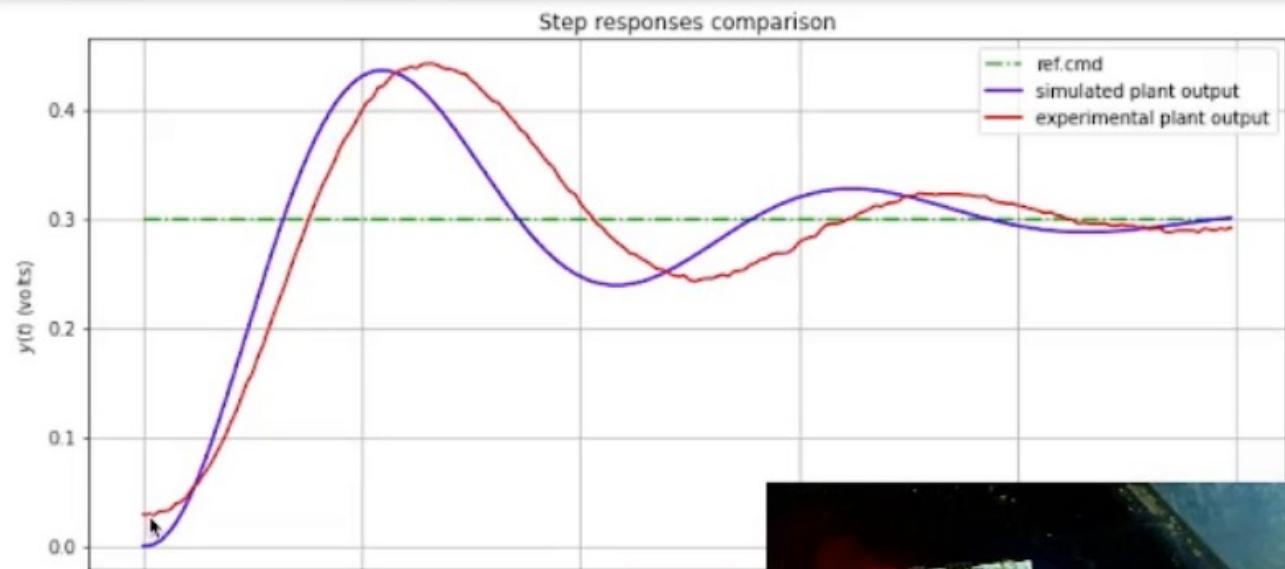
In [35]:

```
datamat = np.array([
[0,0.3,0.02901099,7.330201,3.3],
[0.05,0.3,0.02901099,2.318924,3.3],
[0.1,0.3,0.02813187,0.7021441,2.352144],
[0.15,0.3,0.03164835,0.04646528,1.696465],
[0.2,0.3,0.03164835,-0.04357779,1.606422],
[0.25,0.3,0.03340659,-0.1068736,1.543126],
[0.3,0.3,0.03868132,-0.2249923,1.425008],
[0.35,0.3,0.04043956,-0.1422761,1.507724],
[0.4,0.3,0.04483516,-0.1834713,1.466529],
[0.45,0.3,0.05274725,-0.3039473,1.346053],
[0.5,0.3,0.05714285,-0.2319831,1.418017],
[0.55,0.3,0.06505495,-0.3158633,1.334137],
[0.6,0.3,0.07208791,-0.3195103,1.33049],
[0.65,0.3,0.08087912,-0.3808489,1.269151],
[0.7,0.3,0.09054945,-0.4391118,1.210888],
[0.75,0.3,0.1002198,-0.4712322,1.178768],
[0.8,0.3,0.1125275,-0.5822057,1.067794],
[0.85,0.3,0.1230769,-0.584793,1.065207],
[0.9,0.3,0.1362637,-0.6909042,0.9590958],
[0.95,0.3,0.1494505,-0.7547222,0.8952778],
[1.0,0.3,0.1582417,-0.66117,0.98883],
[1.05,0.3,0.1731868,-0.8482413,0.8017587],
[1.1,0.3,0.1846154,-0.8326234,0.8173765],
[1.15,0.3,0.1995604,-0.9700761,0.6799239],
[1.2,0.3,0.2136264,-1.027014,0.6229861],
[1.25,0.3,0.2268132,-1.055505,0.5944954],
[1.3,0.3,0.24,-1.101719,0.5482807],
```



ผลลัพธ์เพื่อเปรียบเทียบผลตอบสนองขั้นบันได

```
In [38]: # use the same parameters as in the ESP32
kp = 4.8
ki = 2.74
kd = 2.1
pid_compare(kp,ki,kd,datamat,scale=0.3)
```



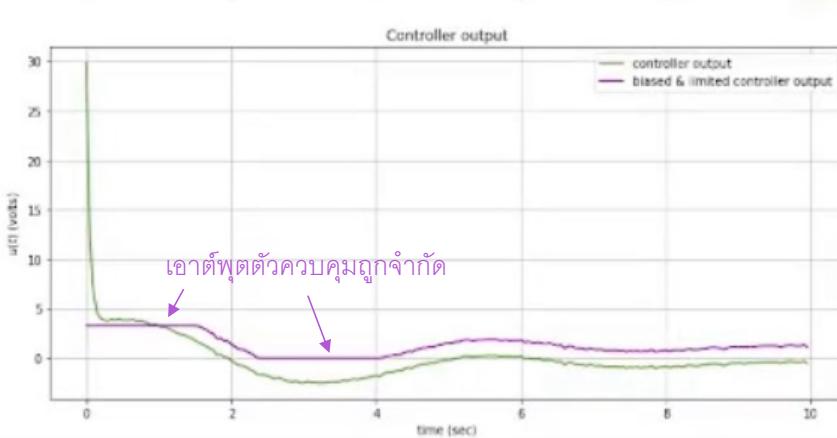
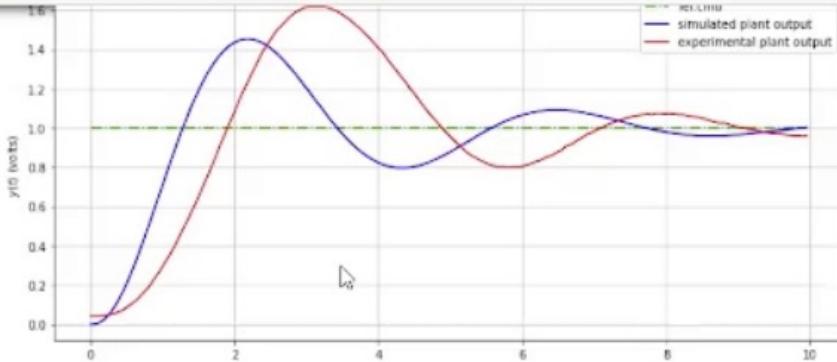
In []:

In []:



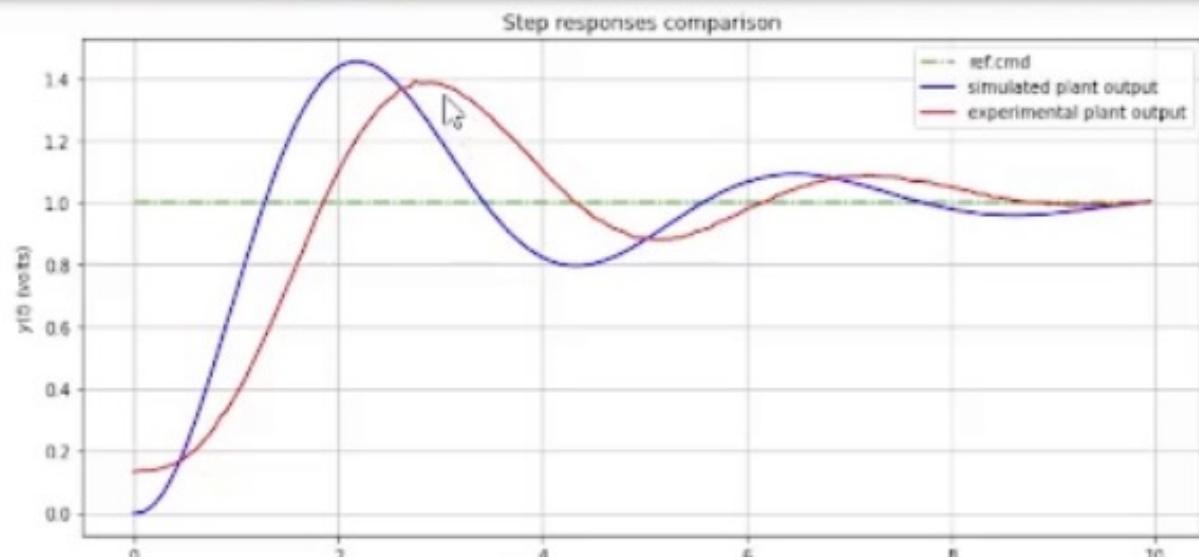
ผลจากการอิมตัว
เมื่อเพิ่มขนาด
 $ref.cmd = 1.0$

```
In [6]: # use the same parameters as in the ESP32
kp = 4.8
ki = 2.74
kd = 2.1
pid_compare(kp,ki,kd,datamat,scale=1)
```



เมื่อใช้ back calculation anti-windup ($kt=0.5$)

```
In [13]: # use the same parameters as in the ESP32  
kp = 4.8  
ki = 2.74  
kd = 2.1  
pid_compare(kp,ki,kd,datamat,scale=1)
```



แบบฝึกหัด

- ทดลองใช้การจำลองผลงานโดยอัลกอริทึมเปรียบเทียบกับการใช้วงจรอิเล็กทรอนิกส์
- ปรับค่า kp , ki , kd อธิบายผลกระทบต่อผลตอบสนองขั้นบันได
- ทดสอบพารามิเตอร์การให้ค่า naïve กับพจน์สัดส่วนและอนุพันธ์

Q&A SESSION

Thank You

