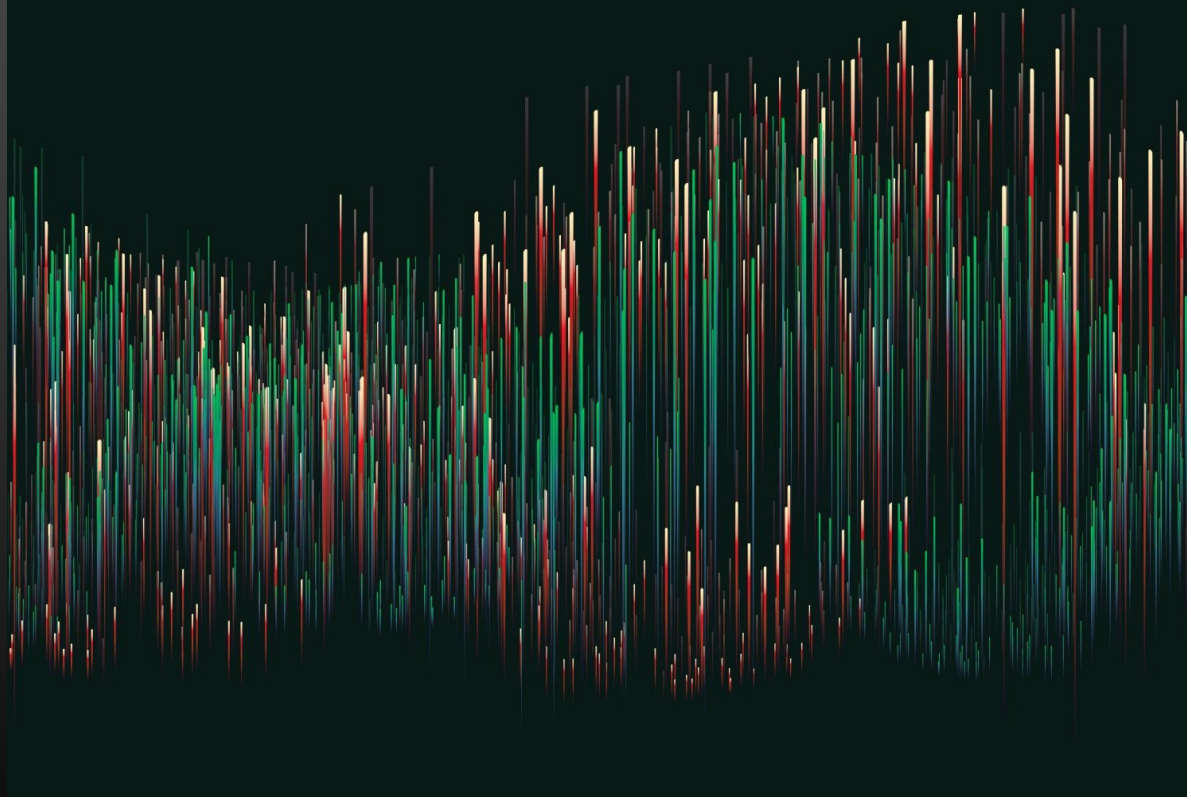
สำหรับการฝึกอบรมเชิงปฏิบัติการ "อุปกรณ์ไอโอทีสำหรับงานควบคุมอุตสาหกรรม"
ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ ม.นเรศวร 26-27 มิถุนายน 2564

การพัฒนา MICROPYTHON บน ESP32

DR.VARODOM TOOCHINDA

DEPT. OF MECHANICAL ENGINEERING

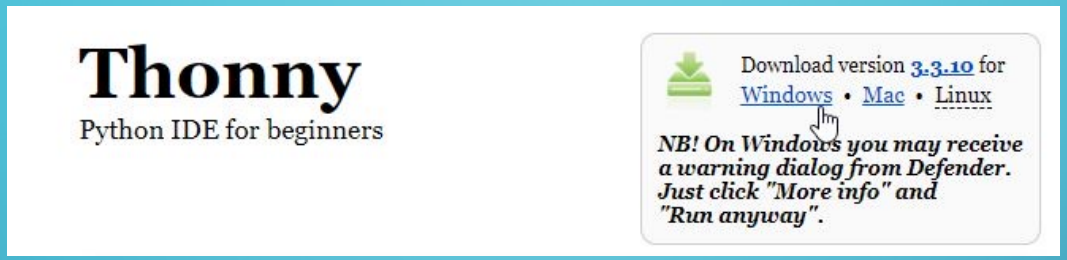
KASETSART UNIVERSITY

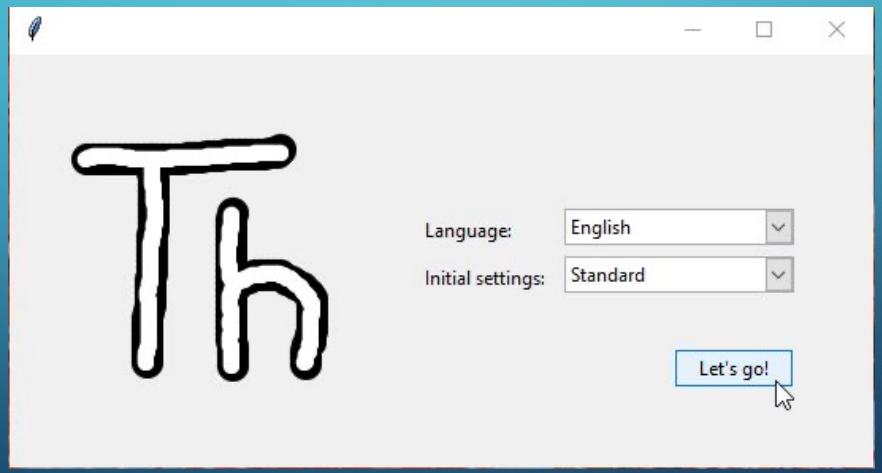


หัวข้อ

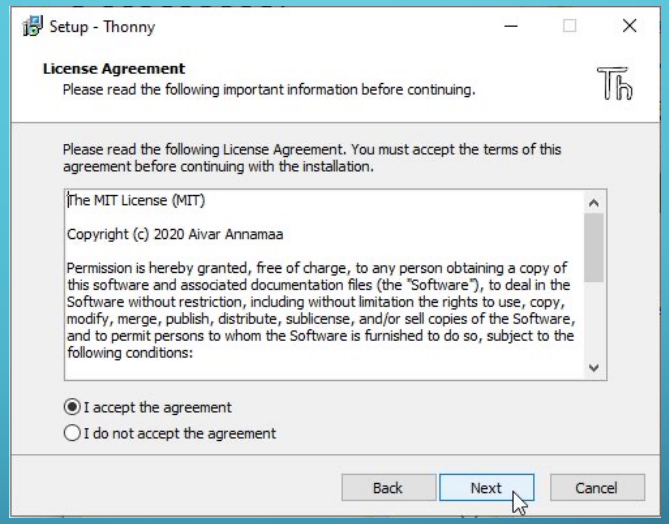
- การใช้งาน Thonny สำหรับพัฒนา micropython บน ESP32
- การเขียนโปรแกรม Python เพื่อใช้งานขาและโมดูลของ ESP32
- การจำลองพลวัตระบบเชิงเส้นบน ESP32
- การเชื่อมต่อกับ NETPIE 2020 โดยแพ็คเกจ umqtt
- การเขียนข้อมูลบน shadow และแสดงผลข้อมูลบน Freeboard
- การส่งคำสั่งจาก NETPIE 2020
- การส่งข้อความไปยัง Freeboard โดยตรงเพื่ออัพเดต
- พัฒนาตัวควบคุมสัดส่วนโดย micropython

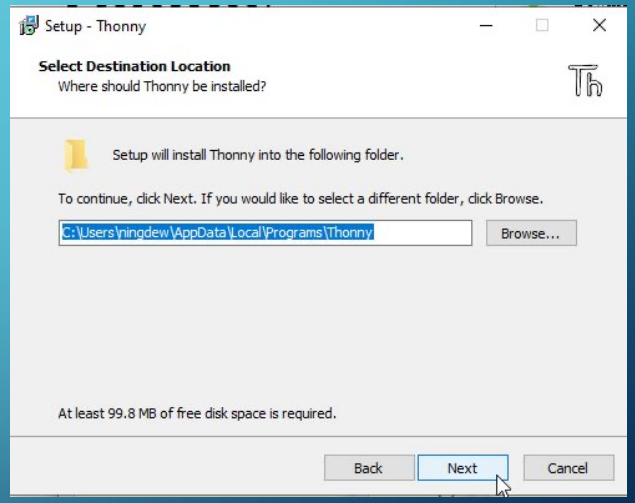
ดาวน์โหลดและติดตั้ง THONNY IDE



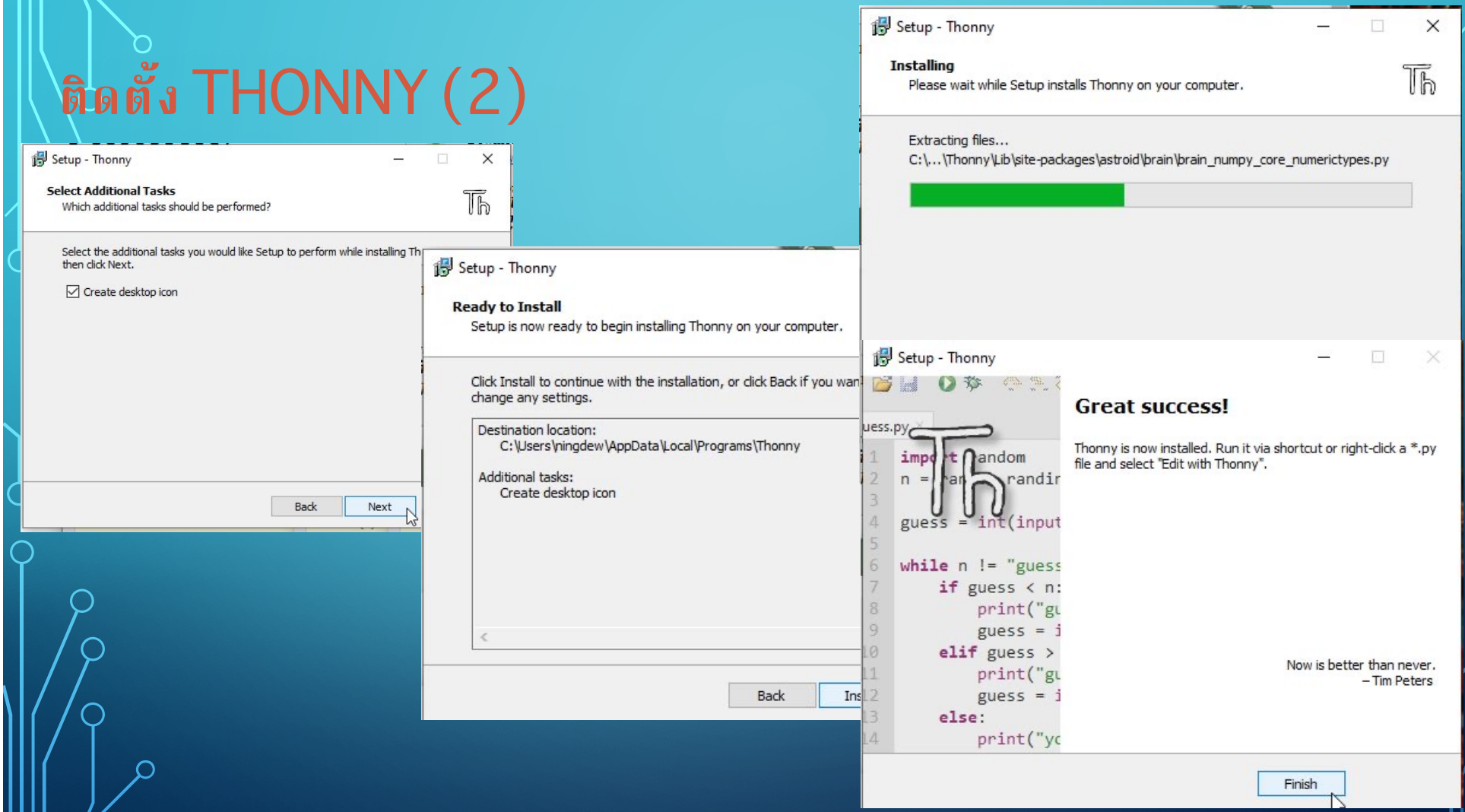


ติดตั้ง THONNY (1)

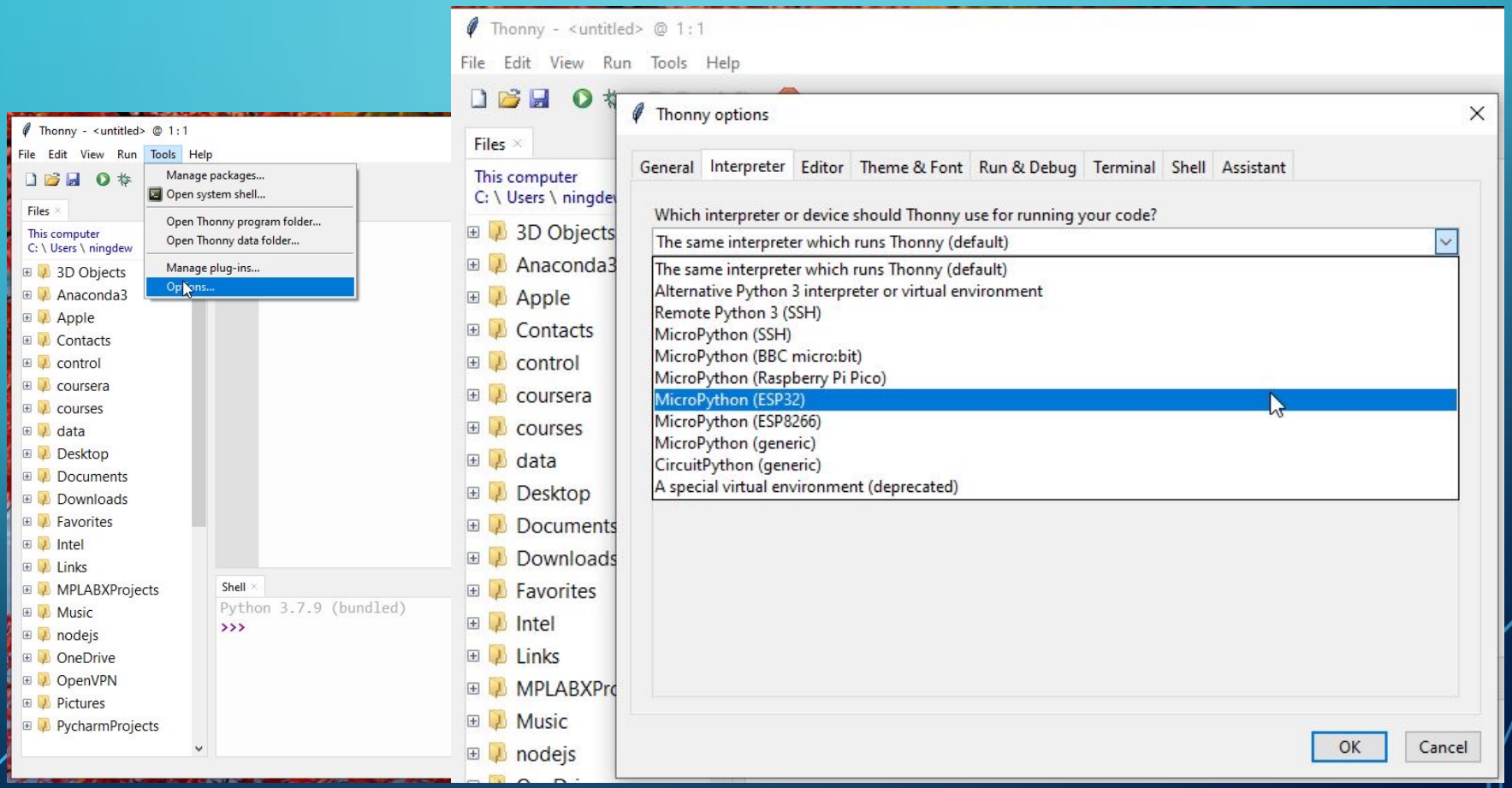




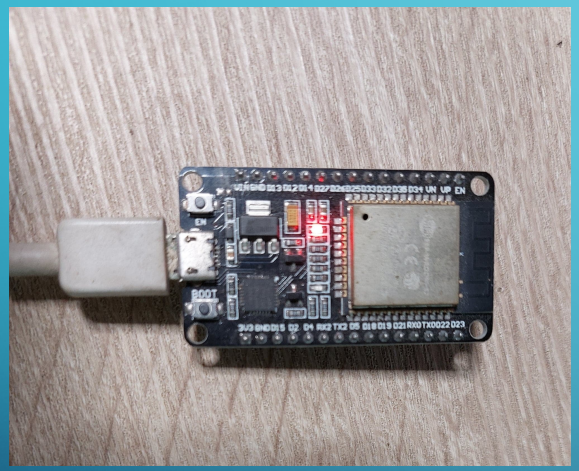
ติดตั้ง THONNY (2)

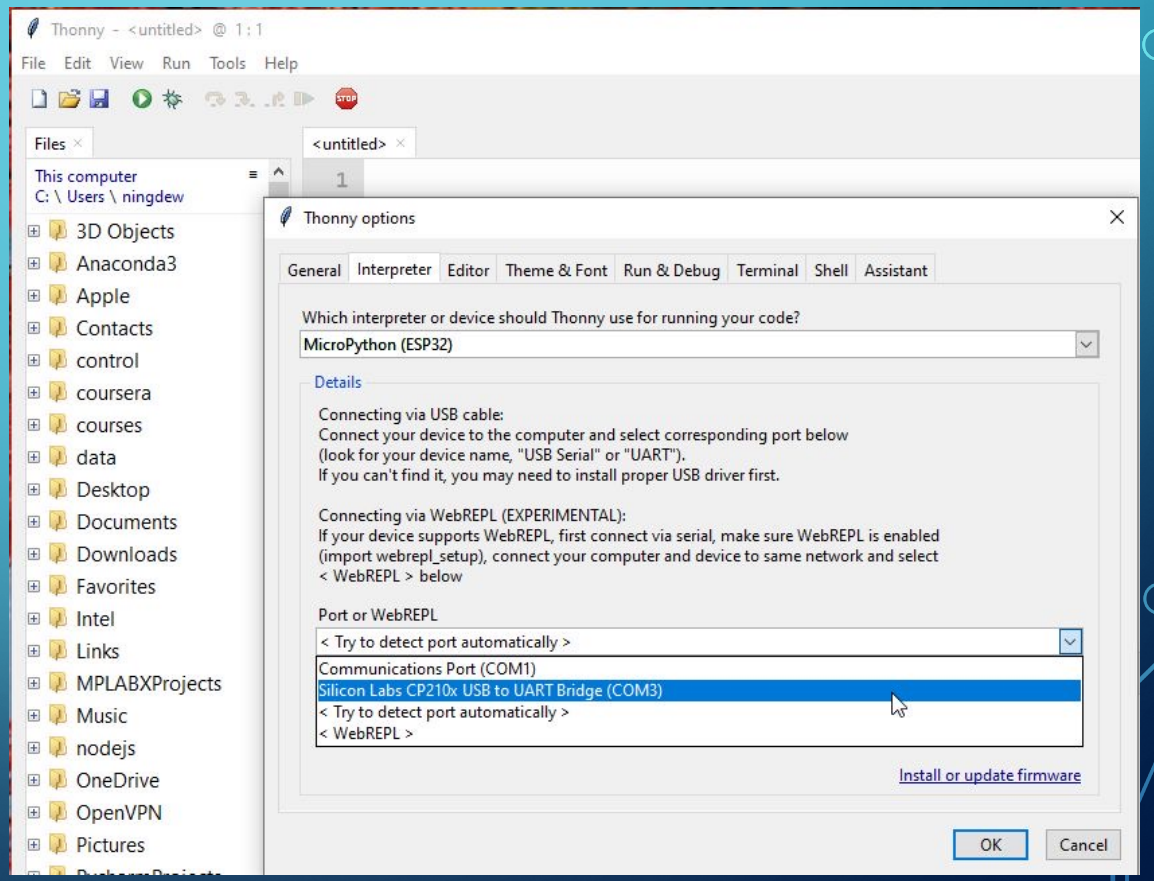


เปิดโปรแกรม THONNY และเลือก OPTION →INTEPRETER

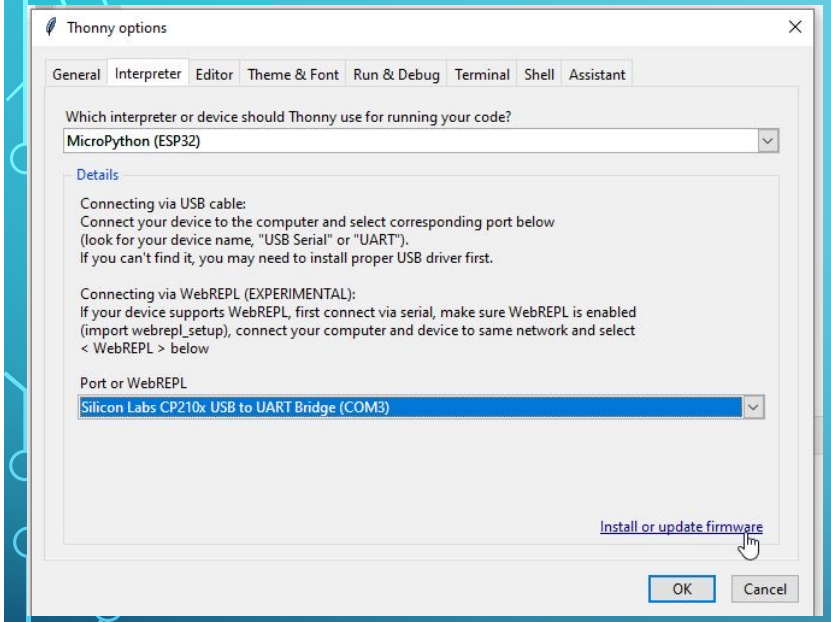


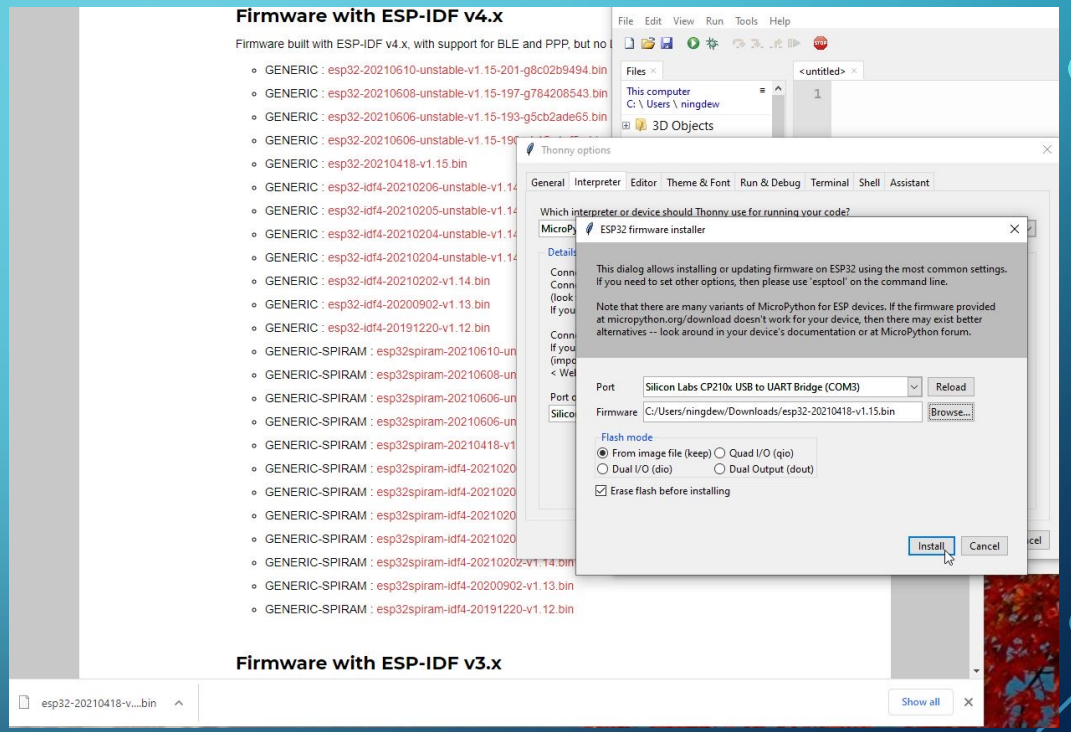
เชื่อมต่อ ESP32 กับพอร์ต USB และติดตั้ง FIRMWARE (1)



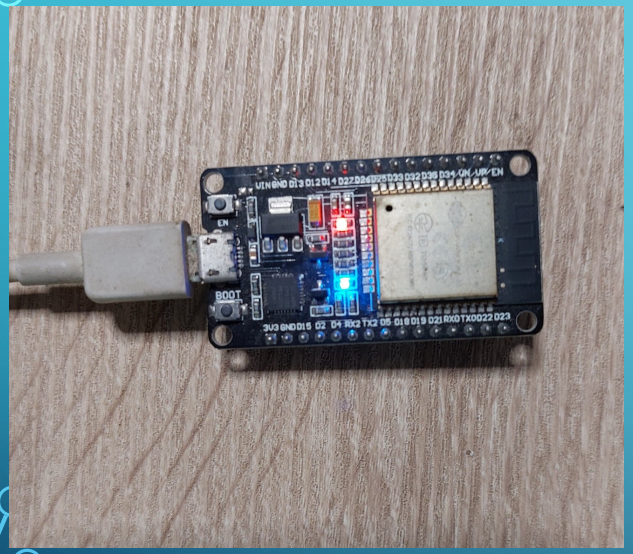


ติดตั้ง FIRMWARE (2)



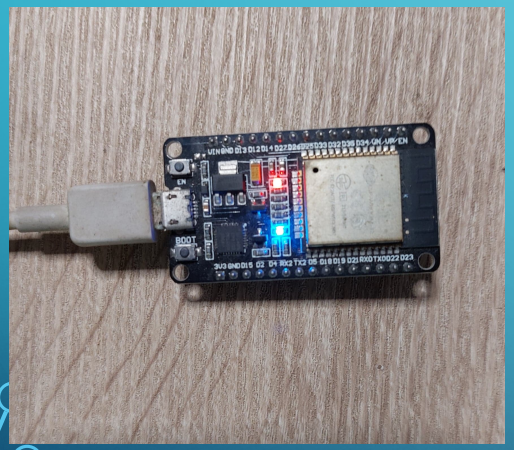


ทดสอบ LED_BLINK.PY



```
from machine import Pin
from utime import sleep_ms

led = Pin(2, Pin.OUT)

while True:
    led.on()
    print("LED is ", led.value())
    sleep_ms(500)
    led.off()
    print("LED is ", led.value())
    sleep_ms(500)
```

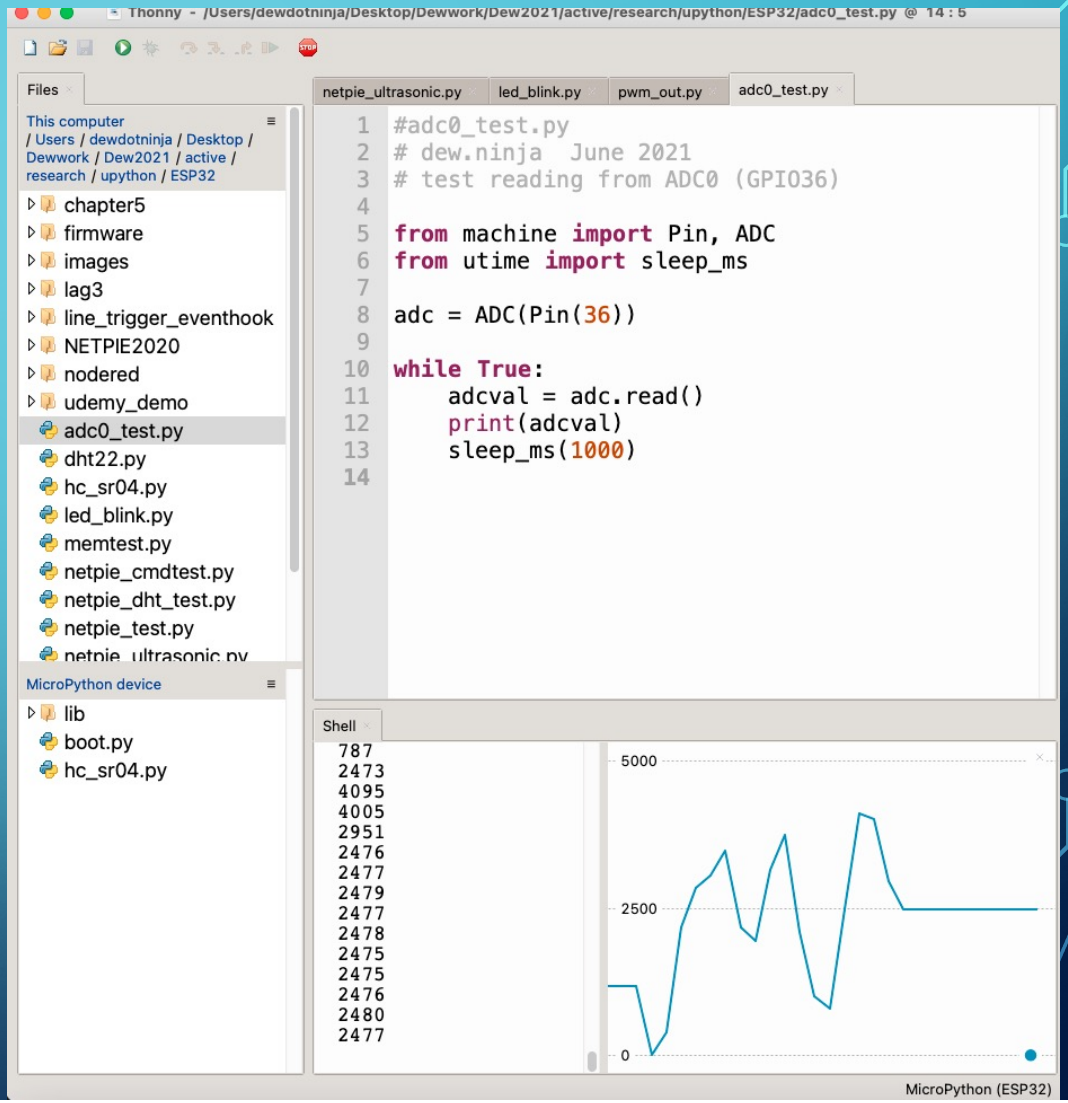
ทดสอบ PWM (PWM_OUT.PY)



```
from machine import Pin, PWM
from utime import sleep_ms
PWMMAX = 1023 # maximum value for PWM output
pwm2 = PWM(Pin(2)) # assign output to pin 2
pwm2.freq(2000)  # set frequency
pwmval = 0  # PWM value
step = 10
direction = 0  # 0 = increase, 1 = decrease

while True:
    if direction==0:
        pwmval += step
    else:
        pwmval -= step
    if pwmval>PWMMAX:
        pwmval = PWMMAX
        direction = 1  # decrease
    elif pwmval<0:
        pwmval = 0
        direction = 0  # increase
    pwm2.duty(pwmval)  # send output
    sleep_ms(10)
```

ทดสอบ ADC (ADC0_TEST.PY)

```
# command to set range 0 - 3.6 volts
adc.atten(ADC.ATTN_11DB)
```



ทดสอบ DAC1 (DAC1_TEST.PY)

```
from machine import Pin, ADC, DAC
from utime import sleep_ms
DACMAX = 255
adc = ADC(Pin(39)) # ADC1_CH3
adc.atten(ADC.ATTN_11DB) # range 0 - 3.6 V
dac1 = DAC(Pin(25))
dac1.write(int(DACMAX/4))
while True:
    adcval = adc.read()
    print(adcval)
    sleep_ms(1000)
```

ทดสอบ TIMER (TIMER_TEST.PY)

```
import machine
from machine import Pin, Timer #importing pin, and timer class
led= Pin(2, Pin.OUT)     # GPIO2 as on-board led

led.value(0)                #LED is off
timer=Timer(-1)

timer.init(period=1000, mode=Timer.PERIODIC, callback=lambda
t:led.value(not led.value()))
```

ทดสอบ Timerโดยเขียนฟังก์ชัน callback (Timer_test2.py)

```
import machine
from machine import Pin, Timer          #importing pin, and timer class
led= Pin(2, Pin.OUT)                  # GPIO2 as on-board led

led.value(0)                  #LED is off
timer1=Timer(1)

def timer_isr(event):
    led.value(not led.value())

timer1.init(period=1000, mode=Timer.PERIODIC, callback=timer_isr)
#initializing the timer
```

ทดสอบเซนเซอร์ DHT (DHT22.PY)

```
import dht
import machine
from utime import sleep_ms
d = dht.DHT22(machine.Pin(4))
while True:

    d.measure()
    temperature=d.temperature()
    humidity=d.humidity()
    print("temperature = ",temperature)
    print("humidity = ",humidity)
    sleep_ms(2000)
```

ทดสอบเซนเซอร์ DHT (DHT22.PY)

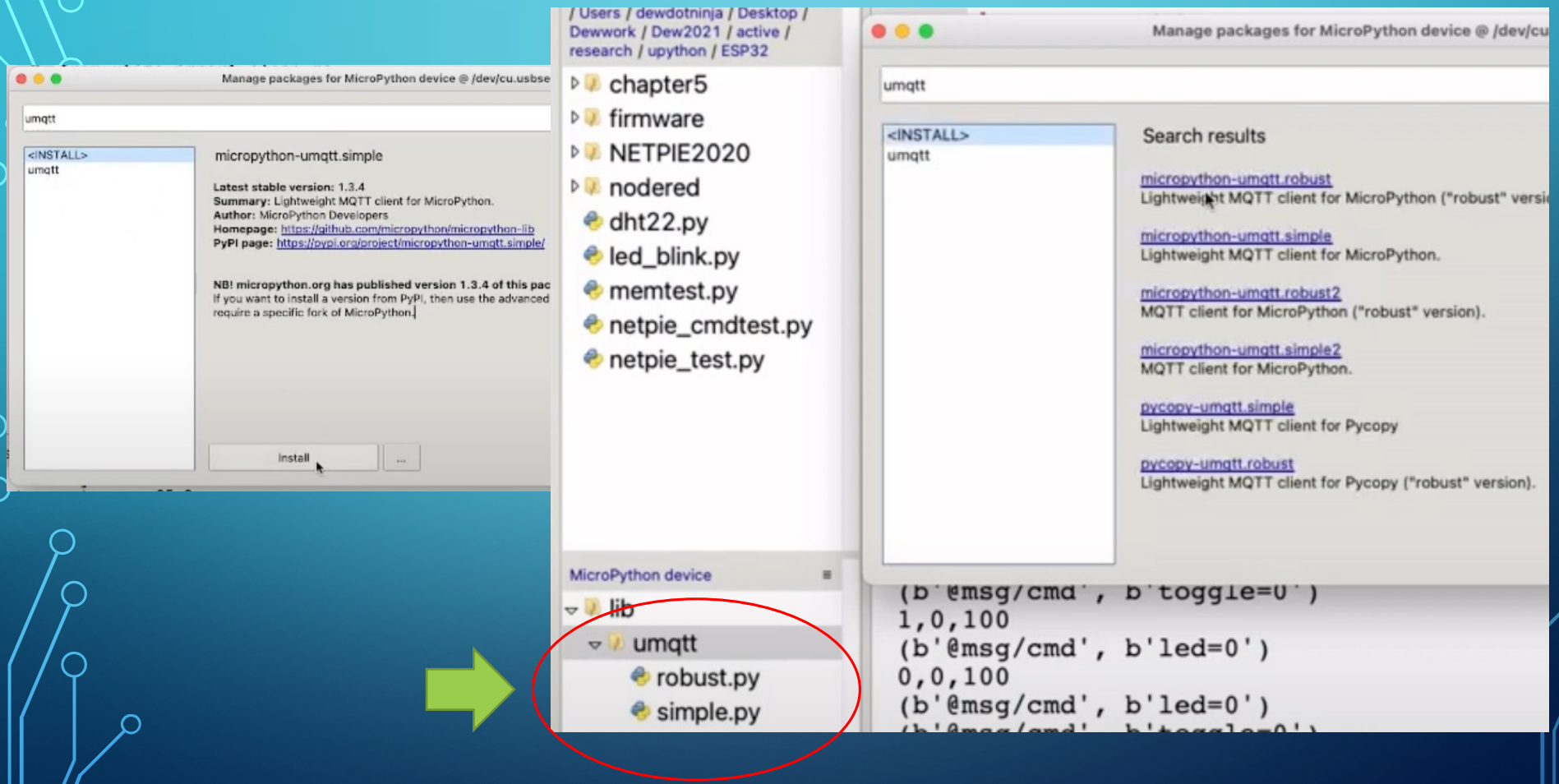
dht22.py | led_blink.py | netpie_test.py

```
1  # dht22.py
2
3  import dht
4  import machine
5  from utime import sleep_ms
6  d = dht.DHT22(machine.Pin(4))
7  while True:
8
9      d.measure()
10     temperature=d.temperature()
11     humidity=d.humidity()
12     print("temperature = ",temperature)
13     print("humidity = ",humidity)
14     sleep_ms(2000)
```

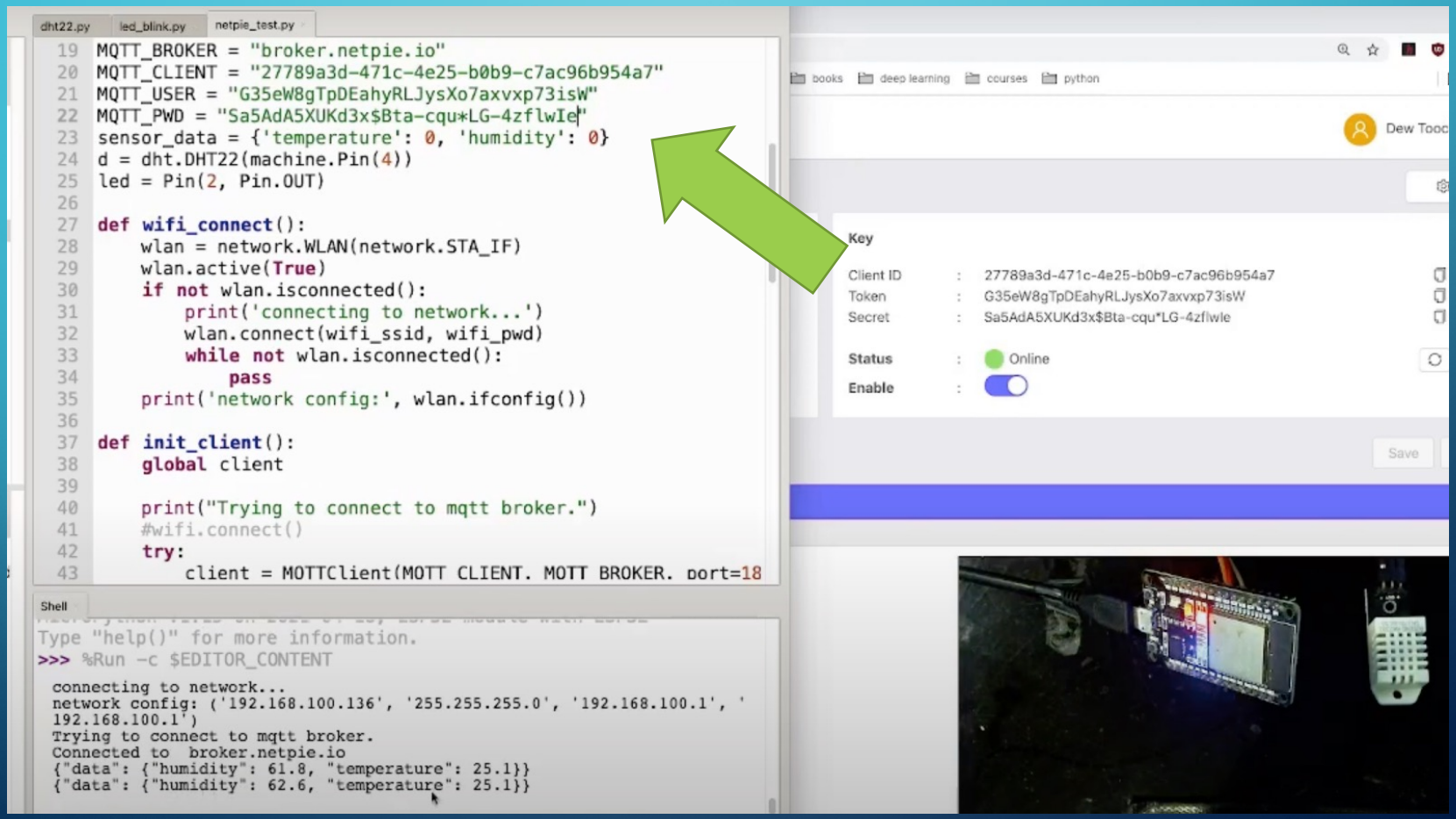
Shell

```
humidity =  61.3
temperature =  25.4
humidity =  61.3
temperature =  25.3
humidity =  61.2
temperature =  25.3
humidity =  61.2
temperature =  25.3
humidity =  61.2
temperature =  25.3
humidity =  61.2
```

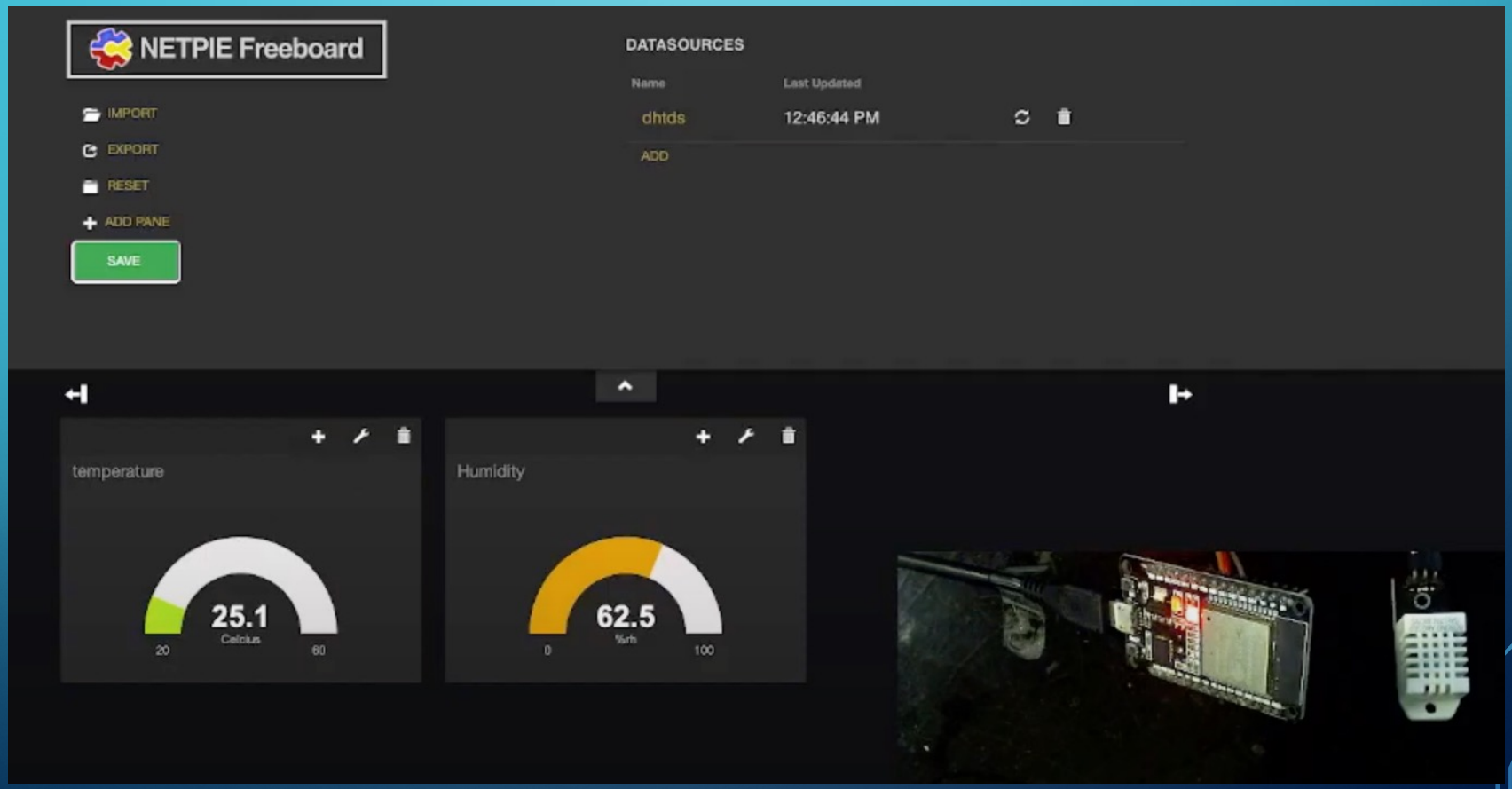
ติดตั้ง MQTT PACKAGE



เชื่อมต่อ DHT กับ NETPIE 2020 (NETPIE_DHT_TEST.PY)



หน้า FREEBOARD แสดงผลอุณหภูมิและความชื้น



พัฒนาส่วนรับคำสั่งจาก NETPIE

```
topic_sub = b"@msg/cmd"
print("Subscribed to ",topic_sub)
client.set_callback(sub_cb)
client.subscribe(topic_sub)
```

เมื่อได้รับคำสั่งหัวข้อ @msg/cmd จะเรียกฟังก์ชัน sub_cb()

```
def sub_cb(topic, msg):
    global cmdtime_current, cmdtime_prev
    print((topic, msg))
    if topic == b'@msg/cmd':
        rcvdstrs = str(msg).split("'") # get rid of b'
        rcvdstr = rcvdstrs[1]
        # this delay is needed for nodered implementation
        cmdtime_current = time.ticks_ms()
        delta_cmdtime = cmdtime_current - cmdtime_prev
        if delta_cmdtime > CMD_DELAY:
            cmdtime_prev = cmdtime_current
            cmdInt(rcvdstr)
```

เรียกฟังก์ชัน cmdInt()

พัฒนาส่วนรับคำสั่งจาก NETPIE

```
def cmdInt(userstr):
    global led_status, toggle_mode, led_period
    result = userstr.find("=")
    if result == -1:
        noparm = 1
        cmdstr = userstr.strip()
    else:
        noparm = 0
        splitstr = userstr.split("=")
        cmdstr = splitstr[0].strip()
        parmstr = splitstr[1].strip()
    #print(cmdstr)
    #print(parmstr)
    if cmdstr.lower() == "led":
        # turn led off/on
    elif cmdstr.lower()== "toggle":
        # set toggle mode
```

แยกส่วนของคำสั่งออกจากพารามิเตอร์ที่เครื่องหมาย =

ทำงานตามคำสั่งที่ได้รับ เช่น ปิด/เปิด led

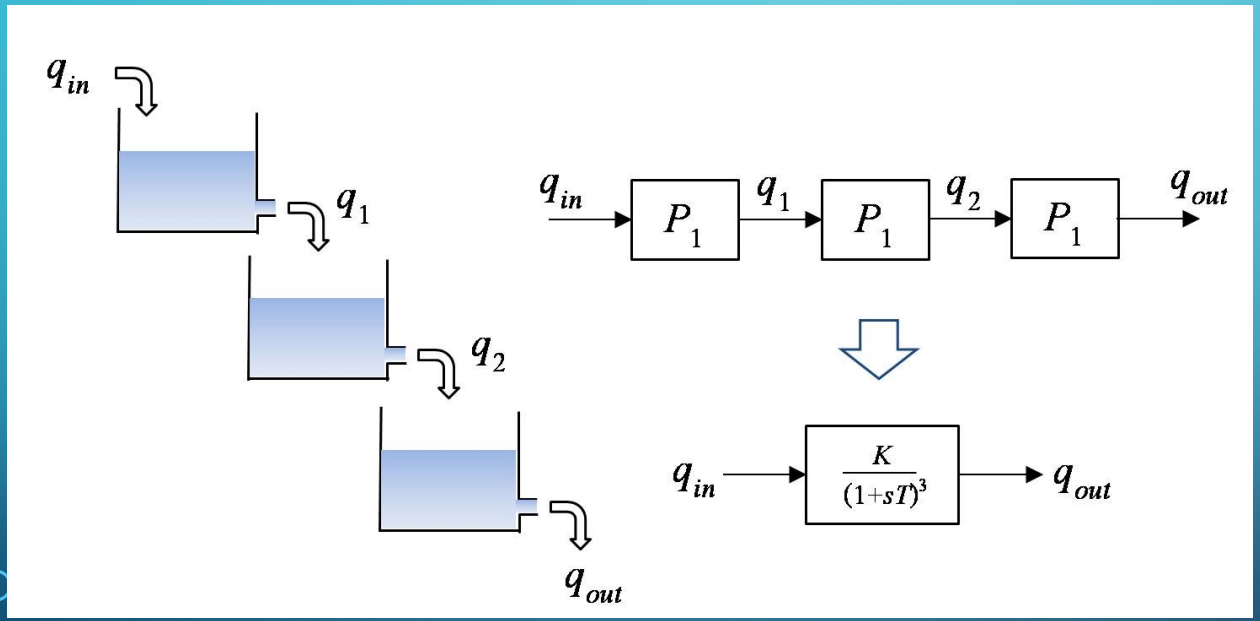
เตรียมพร้อมก่อนทำ EXERCISE 1

- ศึกษาไฟล์โปรแกรม **netpie_cmdtest.py**
- ศึกษาวีดีโอ ส่งคำสั่งจาก NETPIE2020 ไปยังโปรแกรม micropython บน ESP32 จากหน้า micropython github

EXERCISE 1 : ใช้เวลาประมาณ 30 นาที

- พัฒนาโปรแกรมโดย micropython เพื่อสั่งงาน 3 คำสั่งกับ on-board LED
  - led=0/1 : สั่งให้ LED ดับ หรือติดสว่าง ด้วยค่าความสว่างที่กำหนดโดยคำสั่ง brightness
  - toggle=0/1 : สั่งให้ LED กระพริบตามคาบเวลา 200 ms
  - brightness=x : ปรับค่าความสว่างของ LED จาก 0 (ดับ) ถึง 1 (สว่างสุด)
- ใช้เค้าโครงโปรแกรม ex1_netpie_cmdtest.py โดยเติมโค้ดส่วนที่ขาดให้สมบูรณ์
- สร้าง Freeboard เพื่อสั่งงานกับ ESP32 (สามารถ import ไฟล์ dashboard.json)

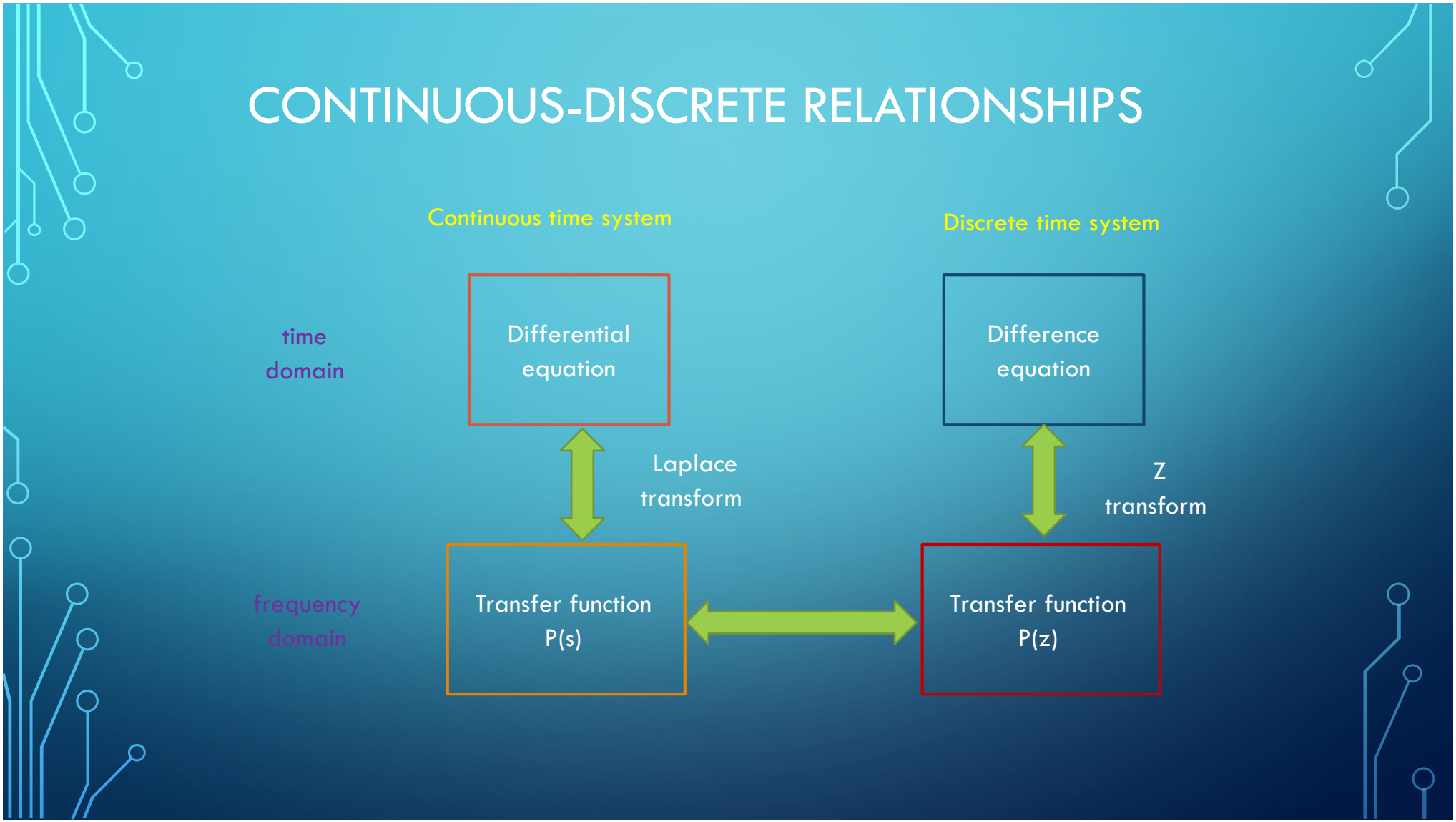
การจำลองพลวัตระบบเชิงเส้นโดยคอมพิวเตอร์



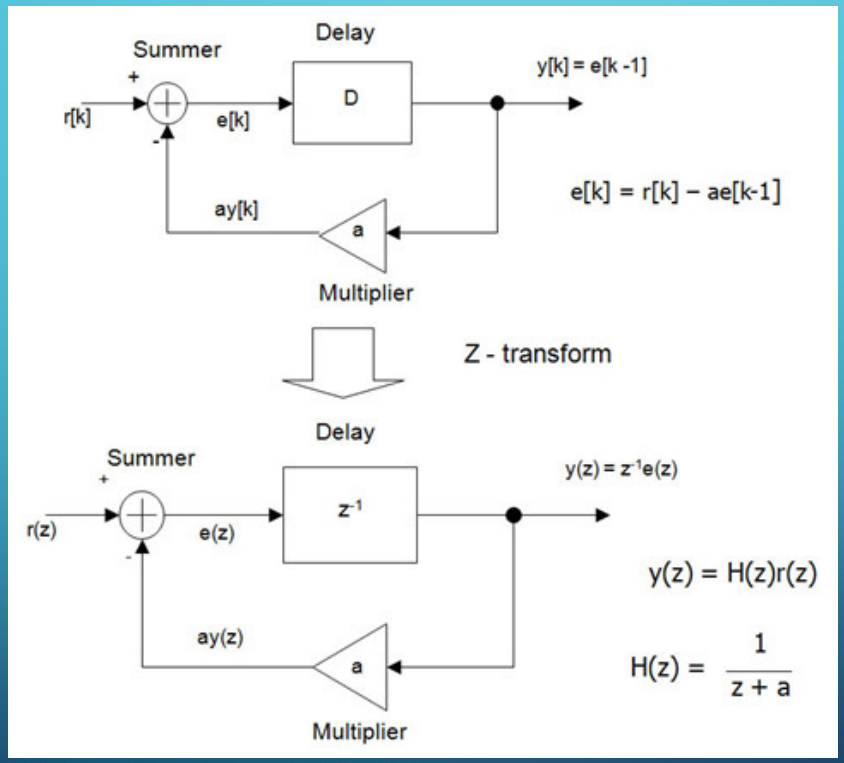
normalized transfer function

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

CONTINUOUS-DISCRETE RELATIONSHIPS



DISCRETE-TIME SYSTEM DESCRIPTION



Difference equation

Discrete-time
Transfer function

TRANSFORMATION OF C(S) TO C(Z)

Forward difference

$$C(z) = C(s)\Big|_{s \to \frac{z-1}{T}}$$

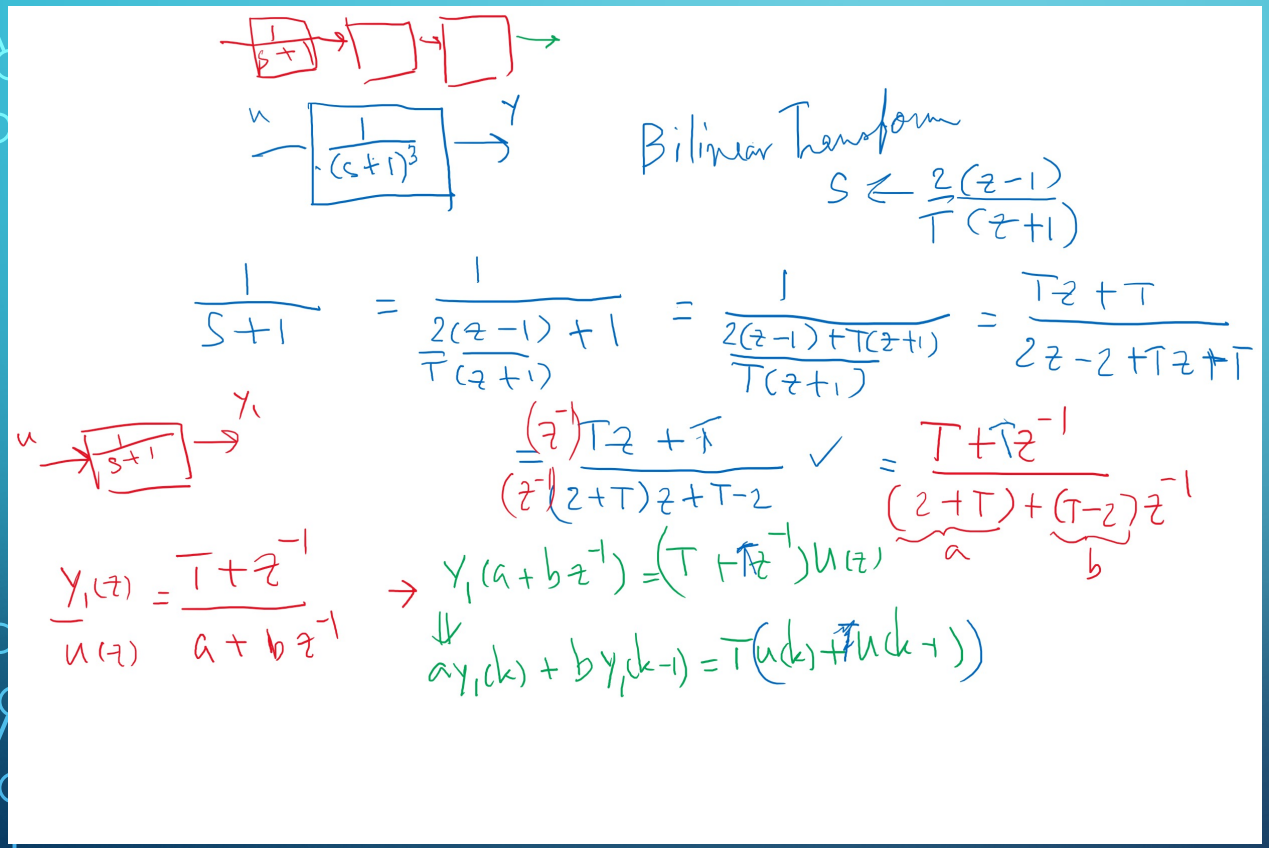
Backward difference

$$C(z) = C(s)\Big|_{s \to \frac{z-1}{Tz}}$$

Bilinear transform

$$C(z) = C(s)\Big|_{s \to \frac{2}{T}\frac{z-1}{z+1}}$$

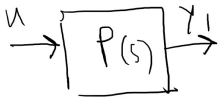
แปลง 1/(S+1) เป็นระบบดีสครีตและเป็นสมการผลต่าง



ดูวีดีโอ การจำลอง 3rd order lag plant โดย micropython บน ESP32

จำลองเบื้องต้นโดยเขียนโค้ดบน JUPYTER NOTEBOOK (1)

$$P(s) = \frac{1}{s+1} \Rightarrow P(z) = \frac{T + z^{-1}}{a + bz^{-1}} \qquad a = (2+T),\ b = (T-2)$$



$$a\,y_1(k) + b\,y_1(k-1) = T(u(k) + u(k-1))$$

$$y_1(k) = \frac{1}{a}\left(-b\,y_1(k-1) + T(u(k) + u(k-1))\right)$$

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

input vectors

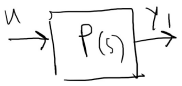
- u_states[0] = previous input of 1st section
- u_states[1] = current input of 1st section
- u_states[2] = previous input of 2nd section
- u_states[3] = current inputt of 2nd section
- u_states[4] = previous input of 3rd section
- u_states[5] = current input of 3rd section

output vectors

- y_states[0] = previous output of 1st section
- y_states[1] = current output of 1st section
- y_states[2] = previous output of 2nd section
- y_states[3] = current output of 2nd section
- y_states[4] = previous output of 3rd section
- y_states[5] = current output of 3rd section

จำลองเบื้องต้นโดยเขียนโค้ดบน JUPYTER NOTEBOOK (2)

$$P(s) = \frac{1}{s+1} \Rightarrow P(z) = \frac{T + z^{-1}}{a + bz^{-1}} \qquad a = (2+T),\ b = (T-2)$$



$$a y_1(k) + b y_1(k-1) = T(u(k) + u(k-1))$$

$$y_1(k) = \frac{1}{a}\left(-b y_1(k-1) + T(u(k) + u(k-1))\right)$$

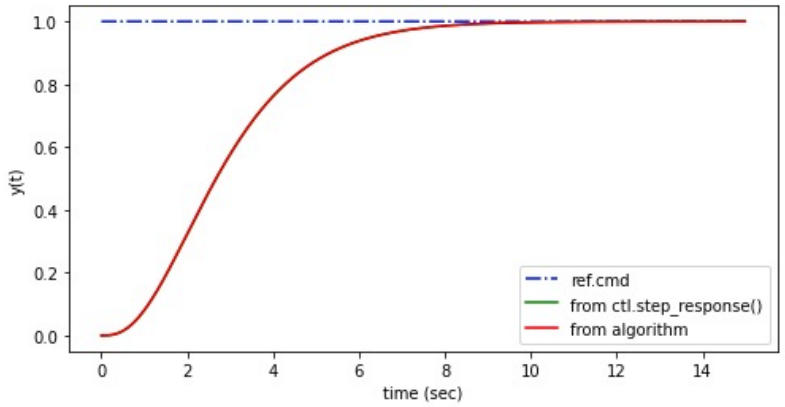
```
# lag3 simulation function
def lag3sim(uvec, tvec):
    yvec = np.zeros(tvec.shape)
    T = tvec[1] - tvec[0]
    a = 2 + T
    b = T - 2
    u_states = np.zeros((6,1))
    y_states = np.zeros((6,1))
    for i in range(len(tvec)):
        yvec[i] = lag3(a,b,T, uvec[i], u_states,
y_states)
    return yvec
```

```
def lag3(a,b,T, u, u_states, y_states):
    for k in range(3):
        y_states[2*k] = y_states[2*k+1]
        u_states[2*k] = u_states[2*k+1]
        if k == 0:
            u_states[2*k+1] = u
        else:
            u_states[2*k+1] = y_states[2*k-1]
        y_states[2*k+1] = (1/a)*(-b*y_states[2*k]+T*(u_states[2*k+1]+u_states[2*k]))
    return y_states[5]
```

จำลองเบื้องต้นโดยเขียนโค้ดบน JUPYTER NOTEBOOK (3)

In [31]:

```
s = ctl.tf('s')
P = 1/(s+1)**3
tvec = np.arange(0,15,0.01)
uvec = np.ones(tvec.shape)
tout, y1 = ctl.step_response(P, tvec)
y2 = lag3sim(uvec, tvec)
plt.figure(figsize=(8,4))
plt.plot(tvec,uvec,'b-.',tvec,y1,'g-',tvec,y2,'r-')
plt.xlabel('time (sec)')
plt.ylabel('y(t)')
plt.legend(['ref.cmd','from ctl.step_response()','from algorithm'])
plt.show()
```



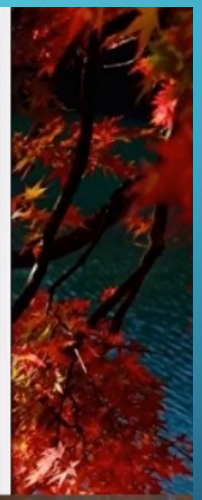
จำลองบน ESP32 (1)

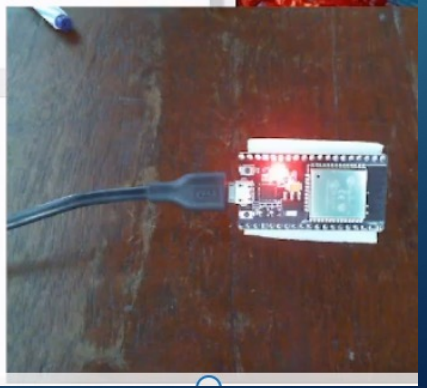
lag3_plantsim.py

```
u = 1

def lag3(a,b,T, u, u_states, y_states):
    for k in range(3):
        y_states[2*k] = y_states[2*k+1]
        u_states[2*k] = u_states[2*k+1]
        if k == 0:
            u_states[2*k+1] = u
        else:
            u_states[2*k+1] = y_states[2*k-1]
        y_states[2*k+1] = (1/a)*(-b*y_states[2*k]+T*(u_states[2*k+1]+u_states[2*k]))
    return y_states[5]


for i in range(datasize):
    if i==0:
        print("datamat = np.array([")

    while dt < T:  # block execution until T expires
        t_current = time.ticks_ms()
        dt = t_current - t_previous

    y = lag3(a,b,T,u, u_states, y_states)
    print("[{},{},{},{}],".format(t,u,y,u))
    if i==datasize-1:
        print("])")
    t+=T
```

Shell

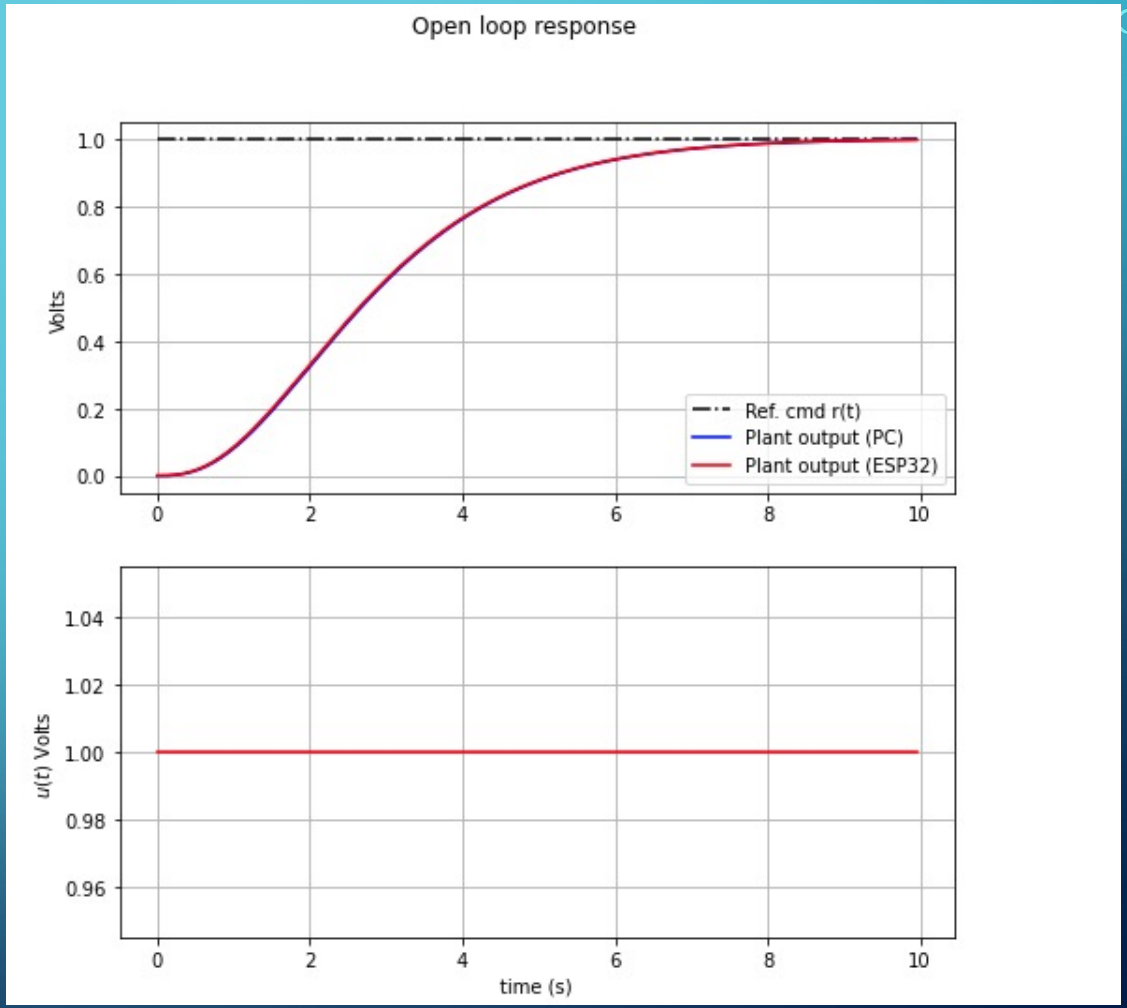
```
  MicroPython v1.15 on 2021-04-18; ESP32 module with ESP32
  Type "help()" for more information.
MicroPython v1.15 on 2021-04-18; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  datamat = np.array([
  [0,1,1.450937e-05,1],
  [0.05,1,9.944225e-05,1],
  [0.1,1,0.0003459548,1],
  [0.15,1,0.0008458681,1],
  [0.2,1,0.001676403,1],
  [0.25,1,0.002901642,1],
  [0.3,1,0.004573871,1],
  [0.35,1,0.006734811,1],
  [0.4,1,0.009416744,1],
  [0.45,1,0.01264355,1],
  [0.5,1,0.01643163,1],
```



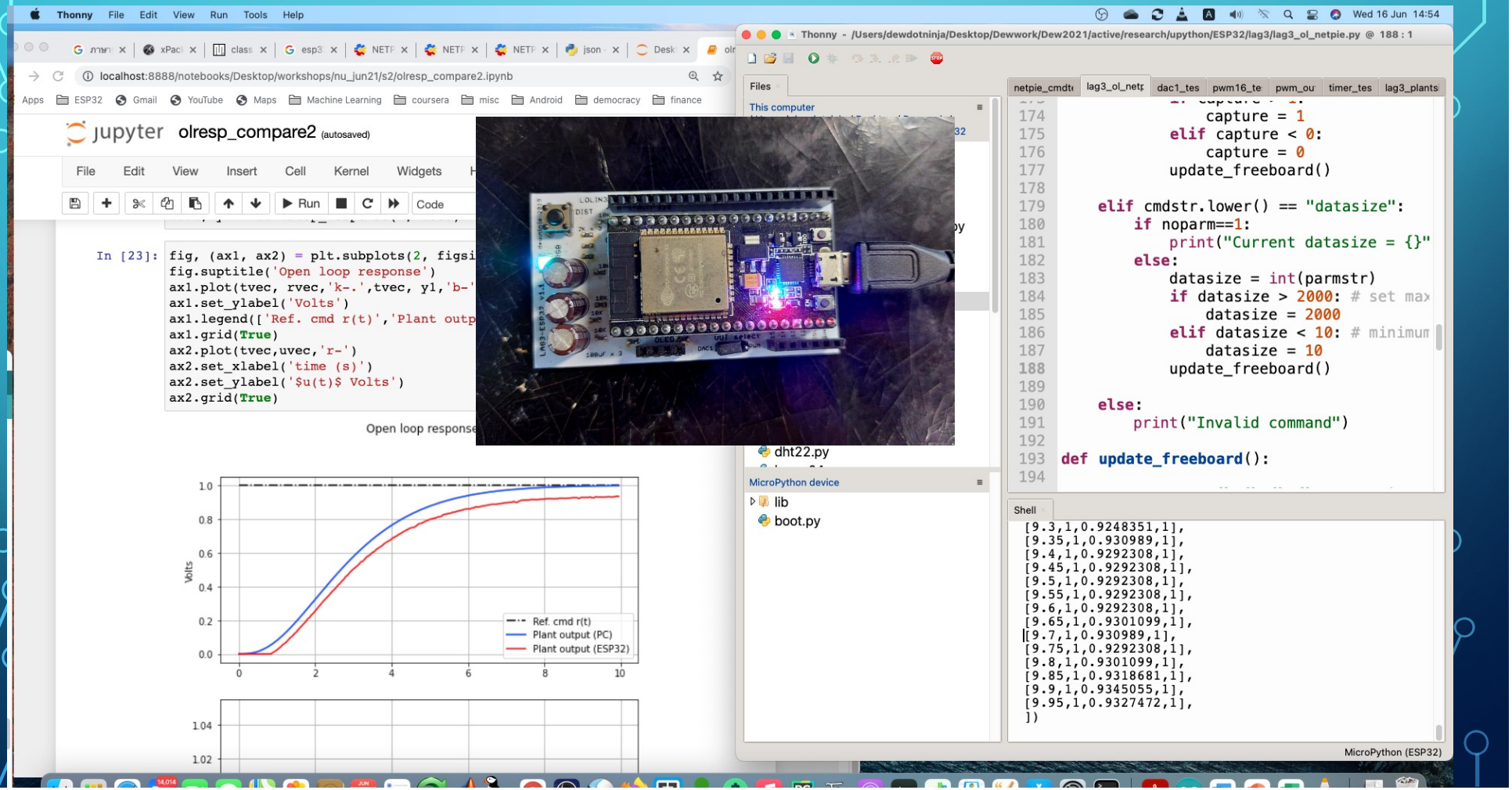


จำลองบน ESP32 (2)

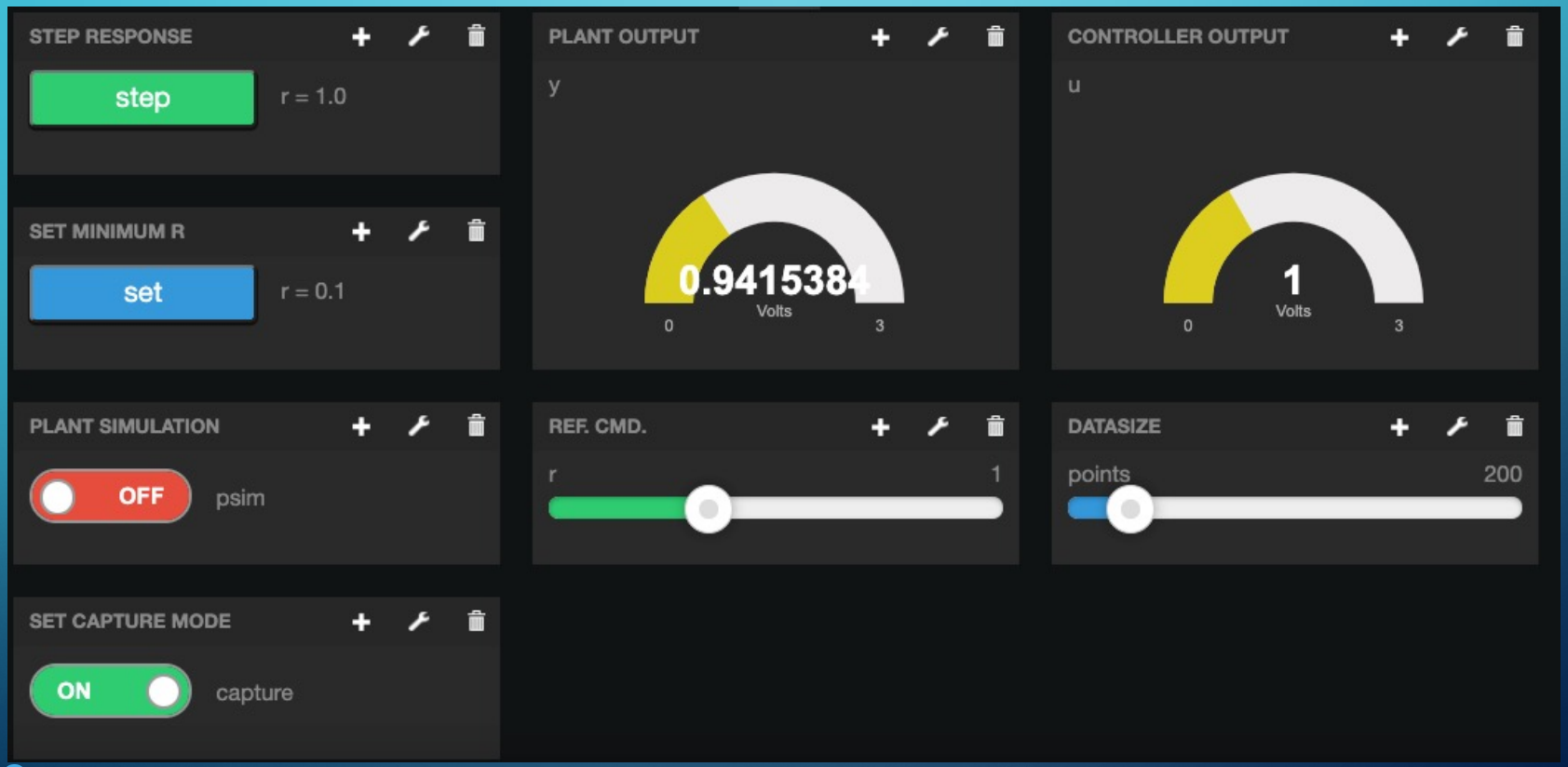
เปรียบเทียบผลตอบสนองระหว่างการจำลองบน PC (Python control library) กับการจำลองบน ESP32



เปรียบเทียบผลตอบสนองวงเปิดของ LAG3

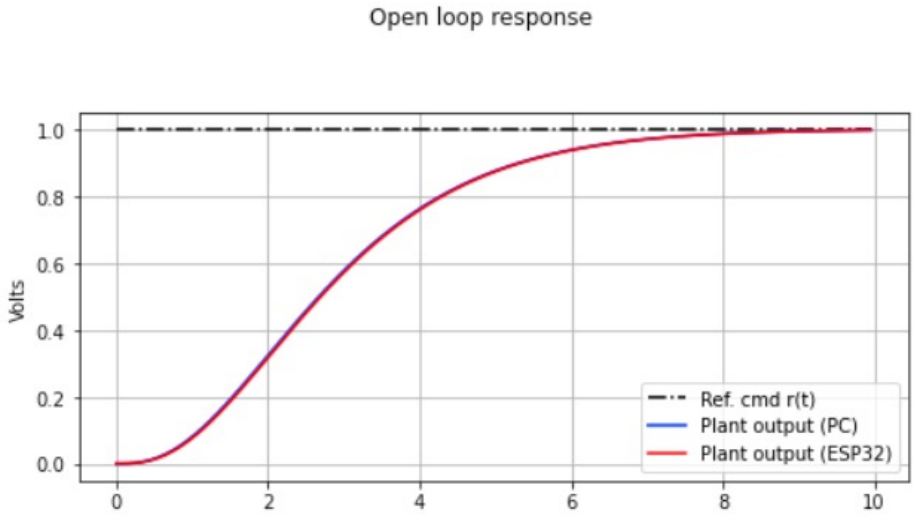


NETPIE FREEBOARD สำหรับ LAG3



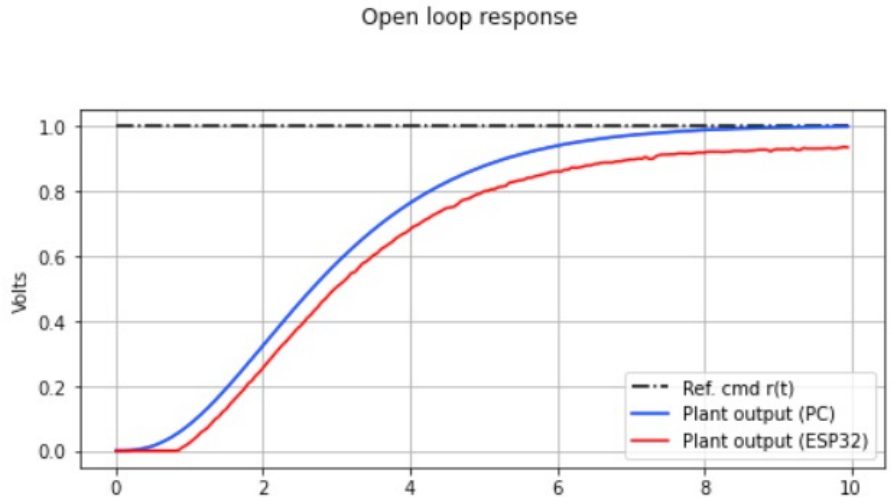
ผลตอบสนองวงเปิดจากการจำลองโดยอัลกอริทึม

In [26]:
```
fig, (ax1, ax2) = plt.subplots(2, figsize=(8,8))
fig.suptitle('Open loop response')
ax1.plot(tvec, rvec,'k-.',tvec, y1,'b-',tvec,yvec,'r-')
ax1.set_ylabel('Volts')
ax1.legend(['Ref. cmd r(t)','Plant output (PC)','Plant output (ESP32)'])
ax1.grid(True)
ax2.plot(tvec,uvec,'r-')
ax2.set_xlabel('time (s)')
ax2.set_ylabel('$u(t)$ Volts')
ax2.grid(True)
```



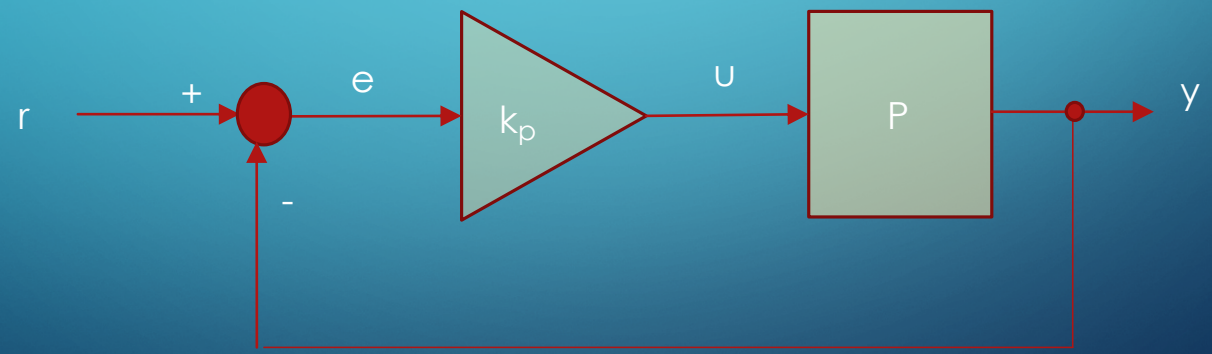
ผลตอบสนองวงเปิดจากการจำลองโดยวงจรอิเล็กทรอนิกส์

```
In [23]: fig, (ax1, ax2) = plt.subplots(2, figsize=(8,8))
         fig.suptitle('Open loop response')
         ax1.plot(tvec, rvec,'k-.',tvec, y1,'b-',tvec,yvec,'r-')
         ax1.set_ylabel('Volts')
         ax1.legend(['Ref. cmd r(t)','Plant output (PC)','Plant output (ESP32)'])
         ax1.grid(True)
         ax2.plot(tvec,uvec,'r-')
         ax2.set_xlabel('time (s)')
         ax2.set_ylabel('$u(t)$ Volts')
         ax2.grid(True)
```



EXERCISE 2 : ใช้เวลาประมาณ 30 นาที

- เพิ่มตัวควบคุม proportional control
- เพิ่ม widget บน Freeboard ดังนี้
  - ปรับค่าพารามิเตอร์ Kp
  - Toggle switch สำหรับสลับค่าระหว่างระบบวงเปิดกับระบบวงปิด



Q&A SESSION

Thank You