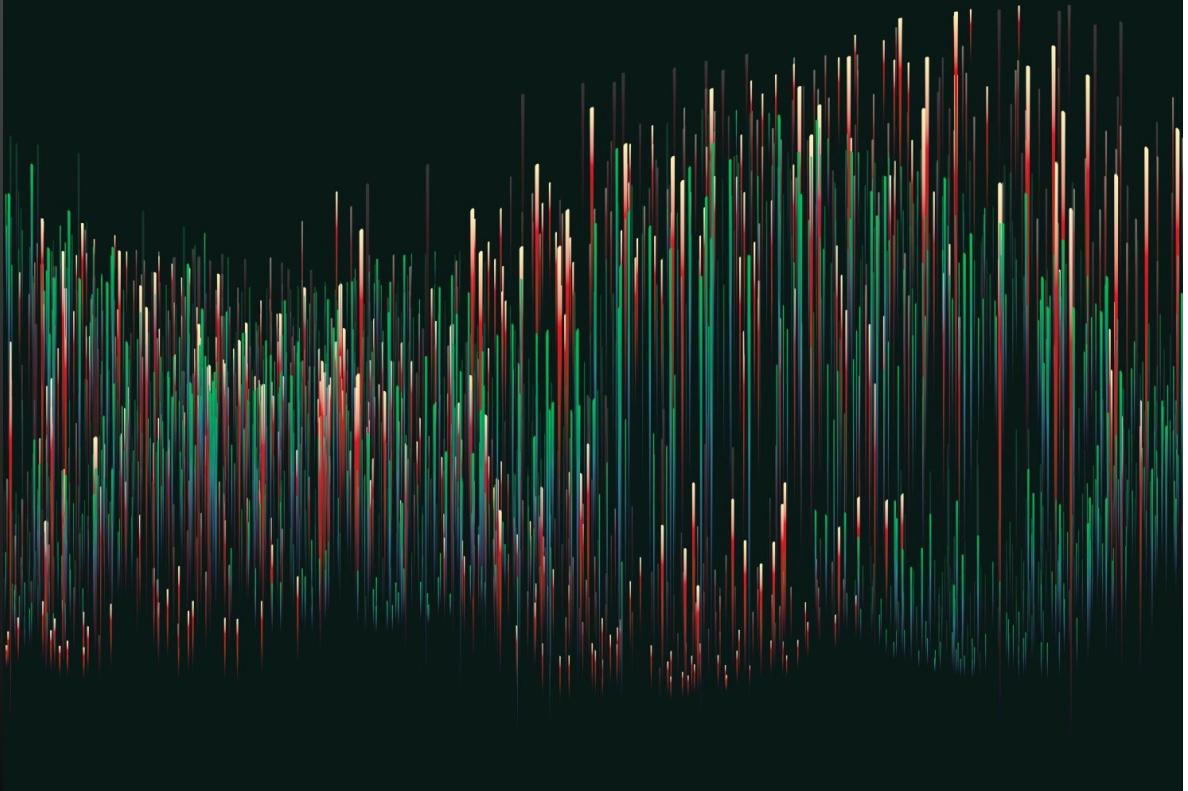




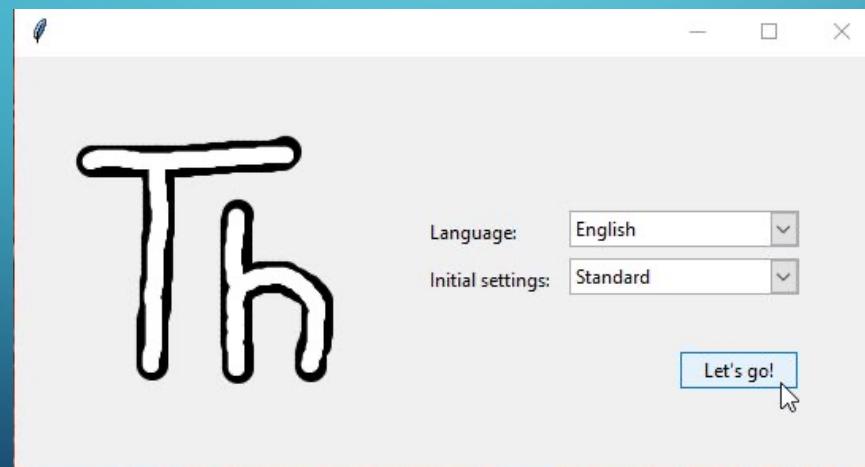
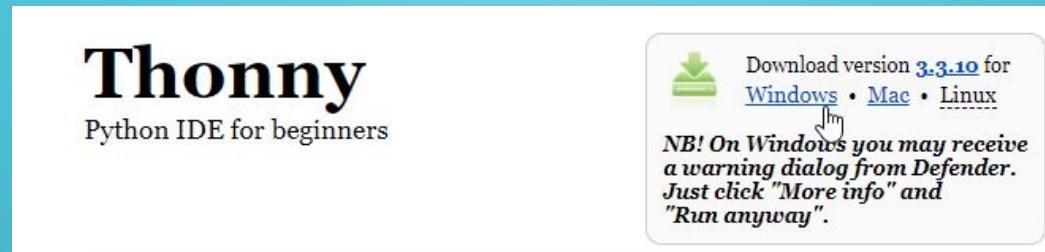
สำหรับการฝึกอบรมเชิงปฏิบัติการ "อุปกรณ์ไอโอทีสำหรับงานควบคุมอุตสาหกรรม"
ภาควิชาพิสิกส์ คณะวิทยาศาสตร์ ม.นเรศวร 26-27 มิถุนายน 2564



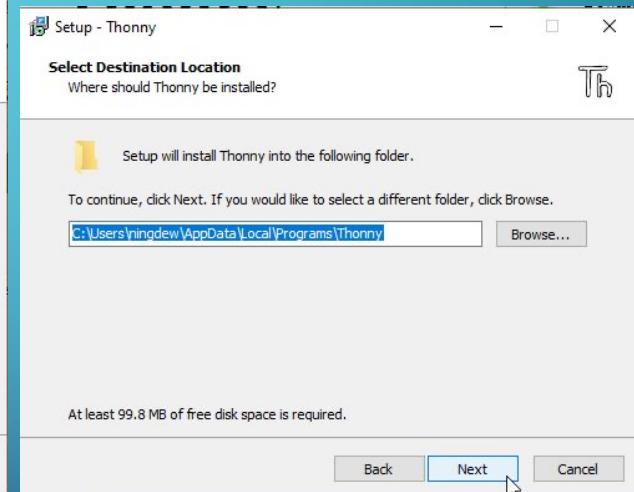
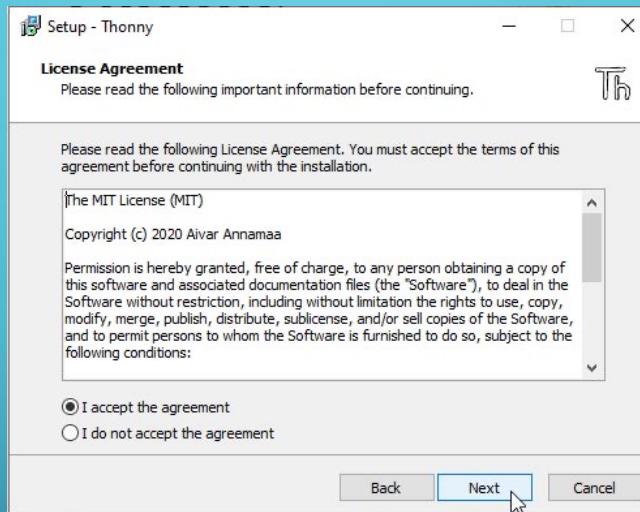
หัวข้อ

- 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
- พัฒนาตัวควบคุมสั่นโดย micropython

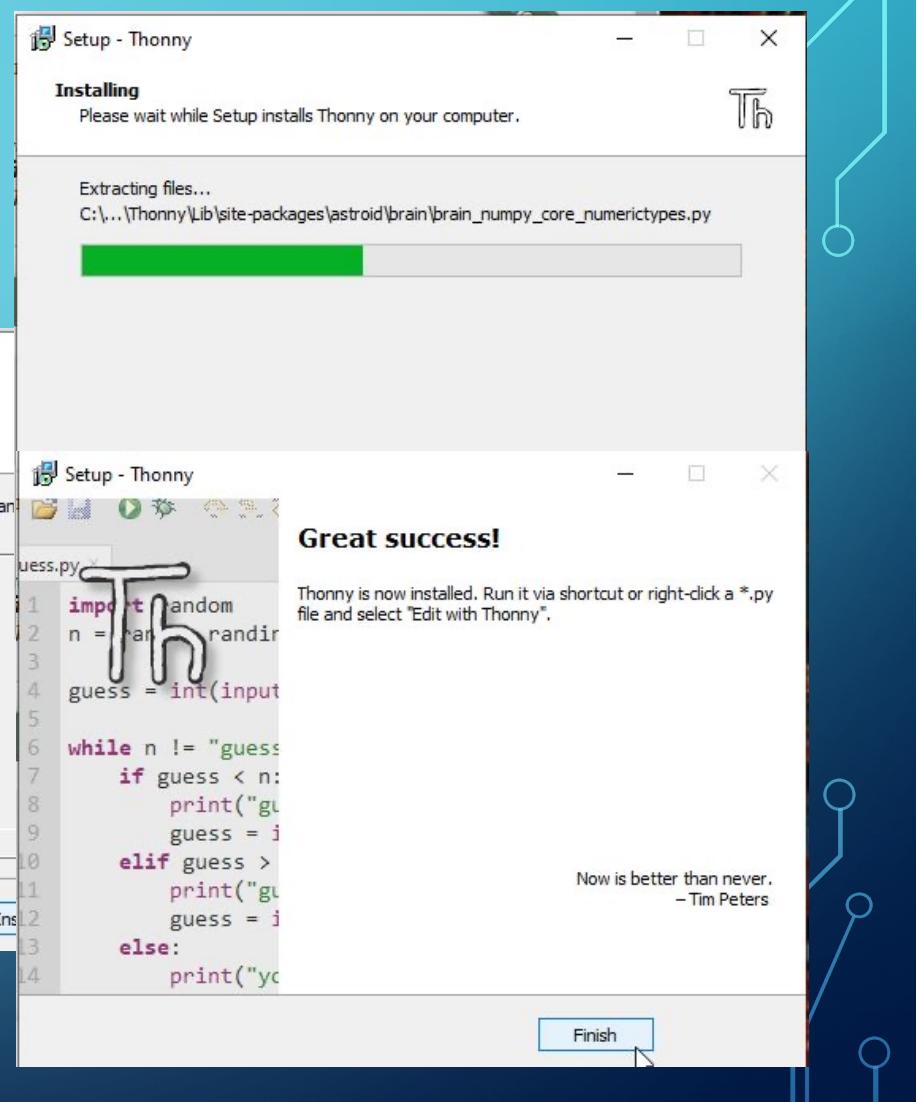
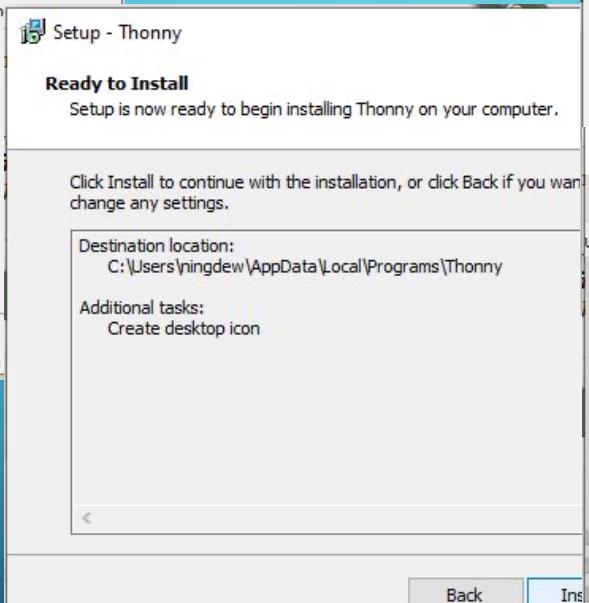
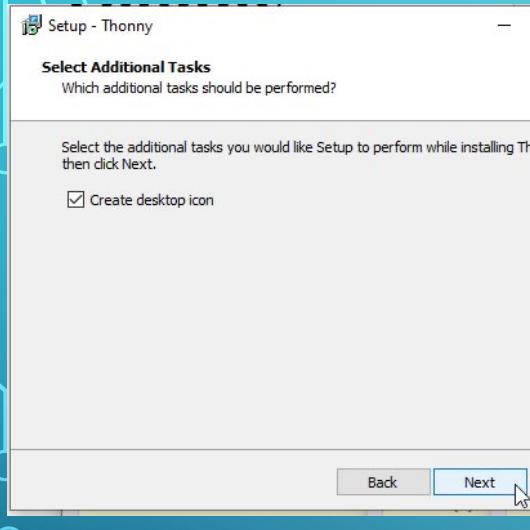
ดาวน์โหลดและติดตั้ง THONNY IDE



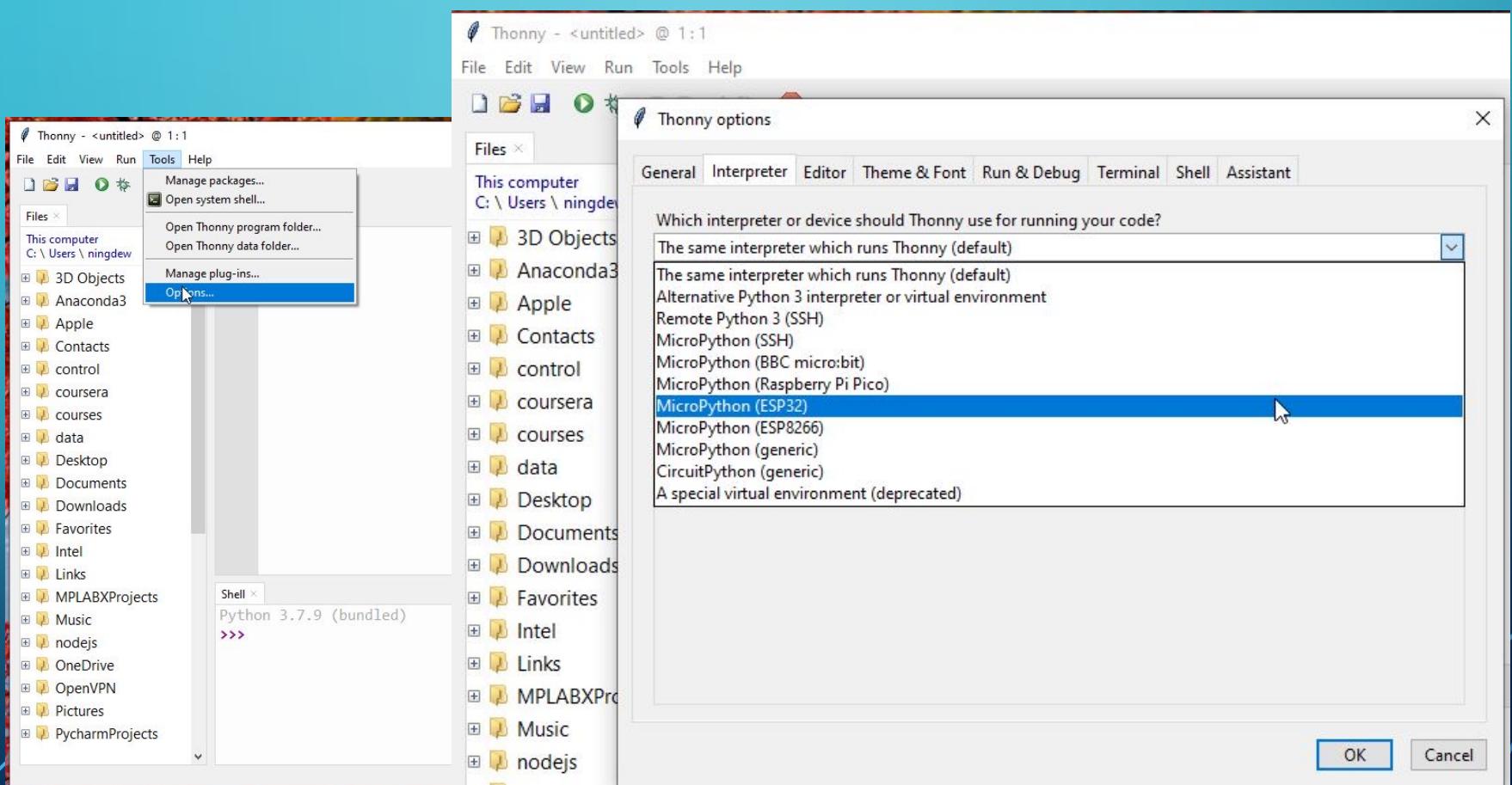
ติดตั้ง THONNY (1)



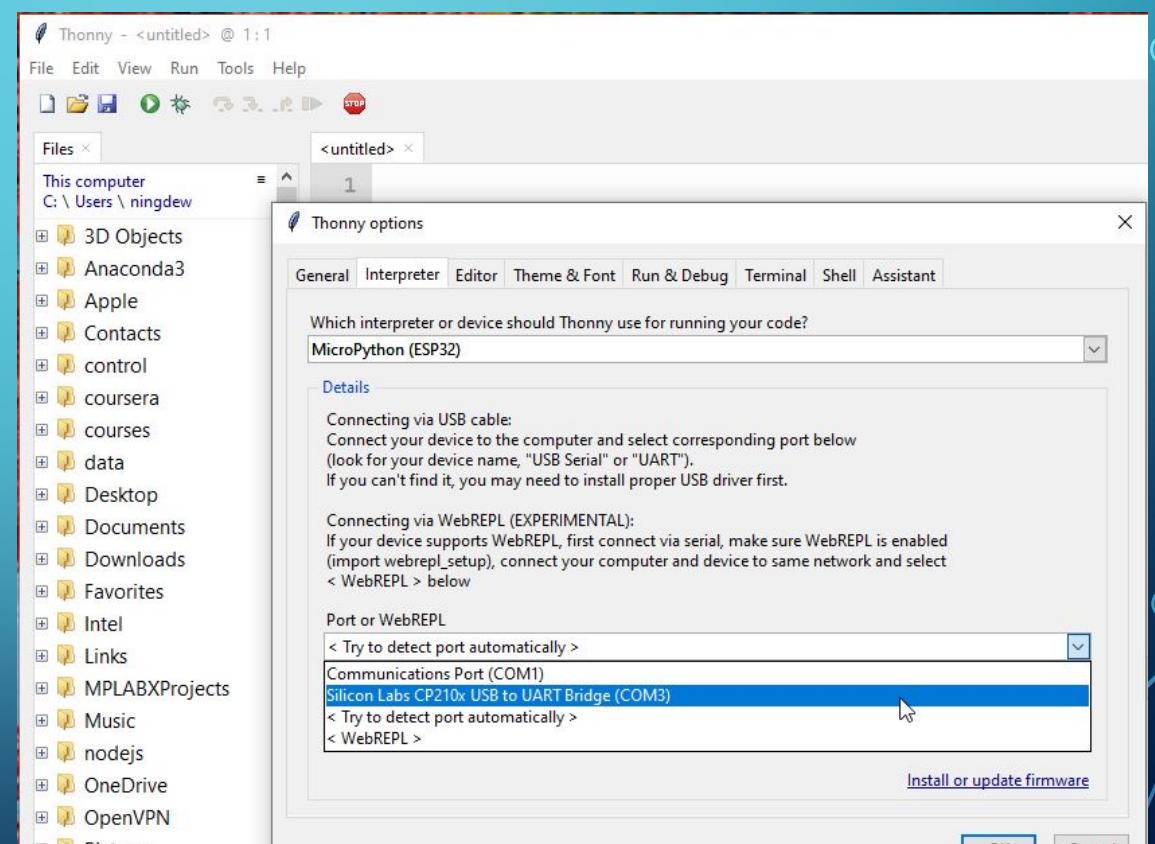
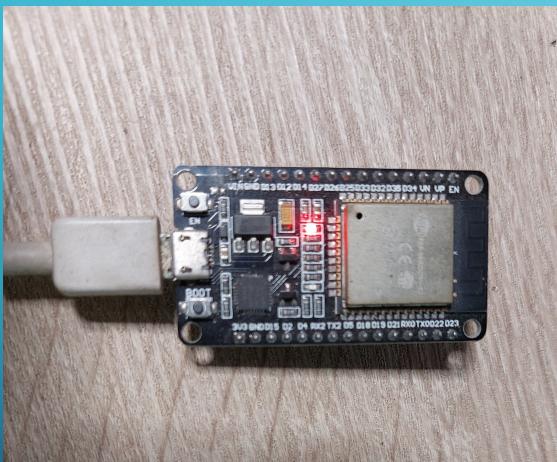
ติดตั้ง THONNY (2)



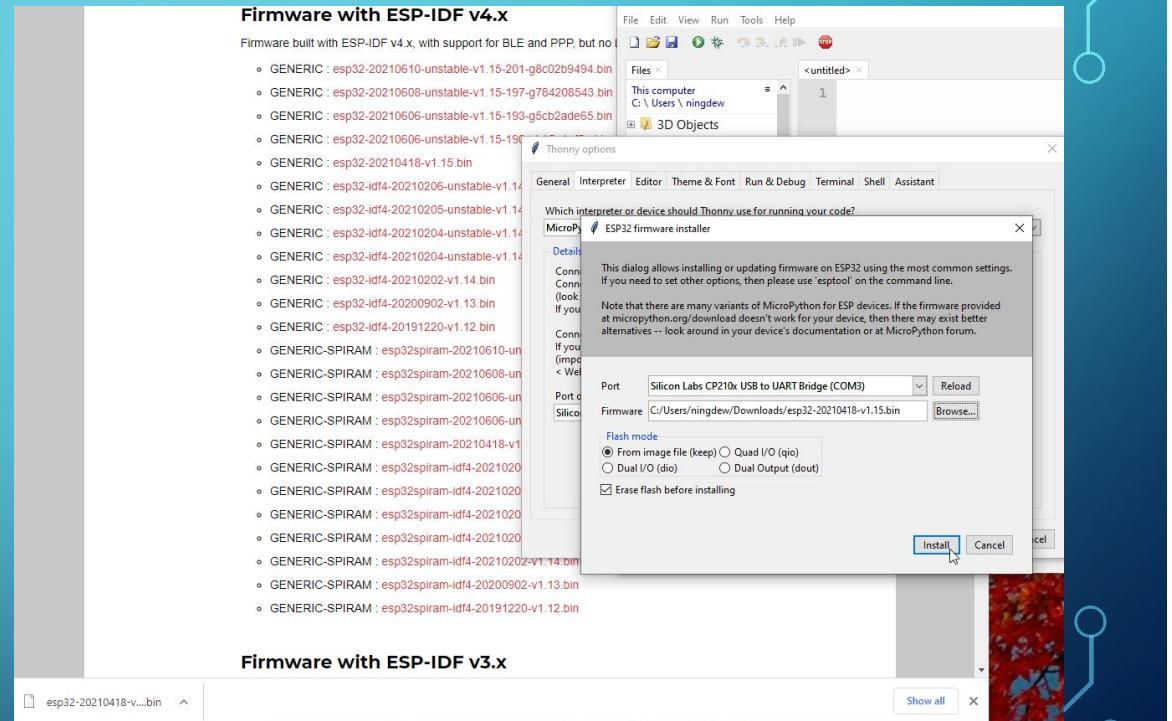
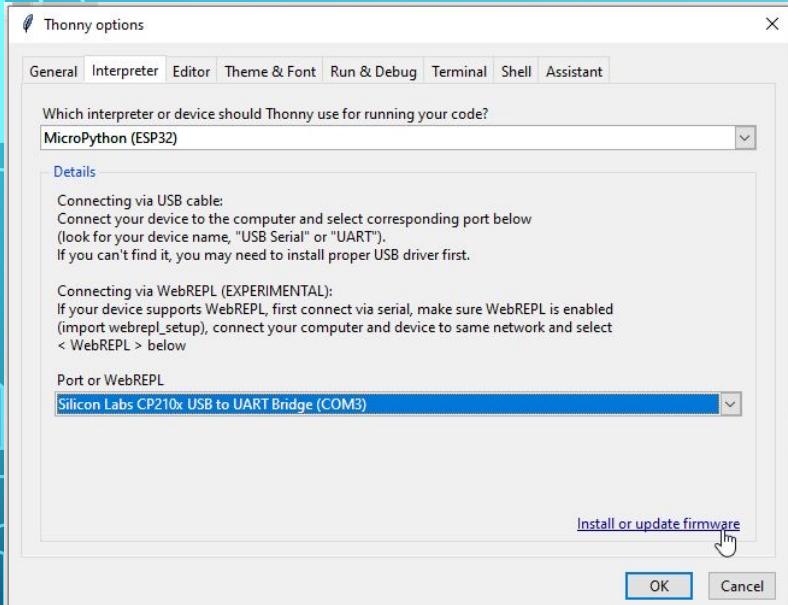
เปิดโปรแกรม THONNY และเลือก OPTION →INTERPRETER



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)
```

The screenshot shows the Thonny IDE interface with the following components:

- Files** tab: Shows a file tree with several Python files and a folder named "MicroPython device".
- Code Editor**: The active tab is "adc0_test.py". The code reads:

```
1 #adc0_test.py
2 # dew.ninja June 2021
3 # test reading from ADC0 (GPIO36)
4
5 from machine import Pin, ADC
6 from utime import sleep_ms
7
8 adc = ADC(Pin(36))
9
10 while True:
11     adcval = adc.read()
12     print(adcval)
13     sleep_ms(1000)
```
- Shell**: Displays a terminal plot of ADC values over time. The x-axis is labeled "MicroPython (ESP32)" and the y-axis ranges from 0 to 5000. The plot shows a noisy signal fluctuating between approximately 2000 and 3000.

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 Timer โดยเขียนฟังก์ชัน 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)
```

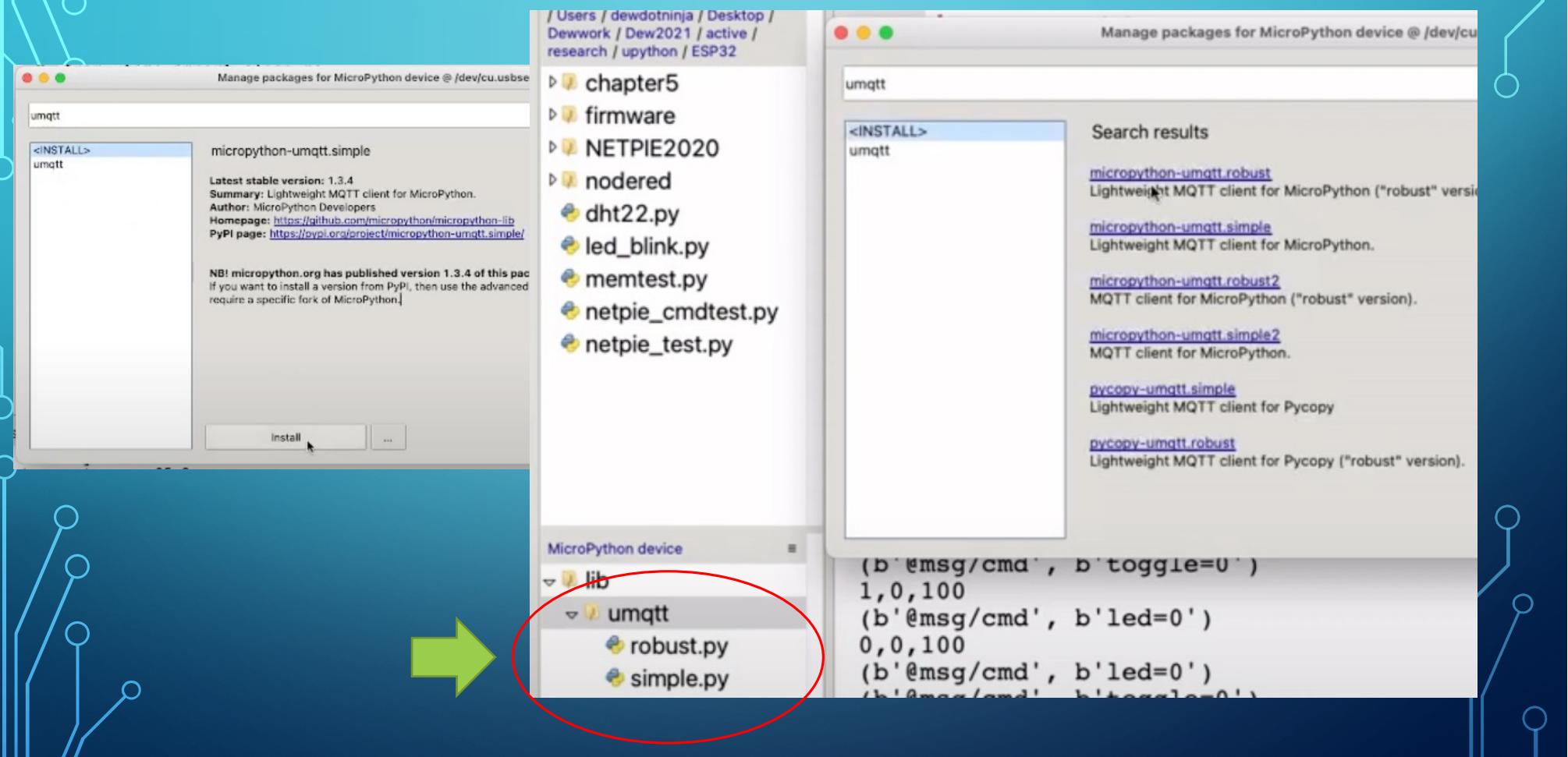
ทดสอบเซนเซอร์ DHT (DHT22.PY)



```
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
```

ติดตั้ง MQTT PACKAGE



เชื่อมต่อ DHT กับ NETPIE 2020 (NETPIE_DHT_TEST.PY)

The screenshot shows a development environment with several windows:

- Code Editor:** Displays the Python script `dht22.py`. The code initializes a DHT22 sensor on pin 4, sets up an LED on pin 2, and connects to a MQTT broker at `broker.netpie.io`. It includes functions for WiFi connection and MQTT client initialization.
- Terminal:** Shows the output of running the script. It prints the network configuration and successfully connects to the MQTT broker, publishing sensor data.
- Netpie Configuration:** A window showing the Netpie configuration interface. It displays the Client ID, Token, and Secret used for authentication. The Status is shown as "Online" with a green indicator, and the Enable switch is turned on.
- Image:** A photograph of the Netpie 2020 hardware, which is a small single-board computer with a display and various pins.

A large green arrow points from the configuration window towards the code editor, indicating the flow of configuration data into the script.

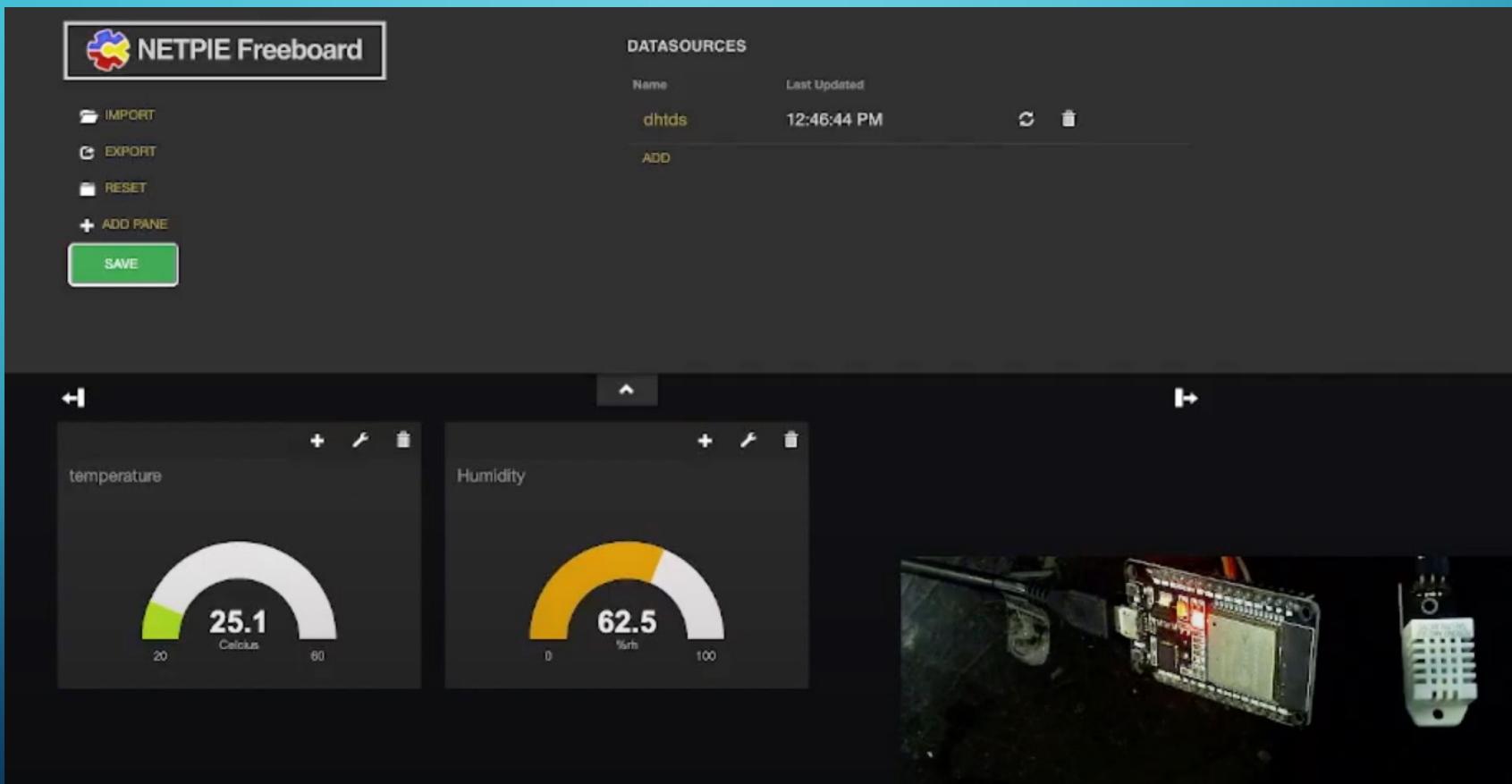
```
MQTT_BROKER = "broker.netpie.io"
MQTT_CLIENT = "27789a3d-471c-4e25-b0b9-c7ac96b954a7"
MQTT_USER = "G35eW8gTpDEahyRLJysXo7axvxp73isW"
MQTT_PWD = "Sa5AdA5XUKd3x$Bta-cqu*LG-4zflwIe"
sensor_data = {'temperature': 0, 'humidity': 0}
d = dht.DHT22(machine.Pin(4))
led = Pin(2, Pin.OUT)

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())

def init_client():
    global client

    print("Trying to connect to mqtt broker.")
    #wifi.connect()
    try:
        client = MOTTClient(MOTT_CLIENT, MOTT_BROKER, port=18
Shell
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
connecting to network...
network config: ('192.168.100.136', '255.255.255.0', '192.168.100.1', '192.168.100.1')
Trying to connect to mqtt broker.
Connected to broker.netpie.io
{"data": {"humidity": 61.8, "temperature": 25.1}}
{"data": {"humidity": 62.6, "temperature": 25.1}}
```

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:
        nparm = 1
        cmdstr = userstr.strip()
    else:
        nparm = 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

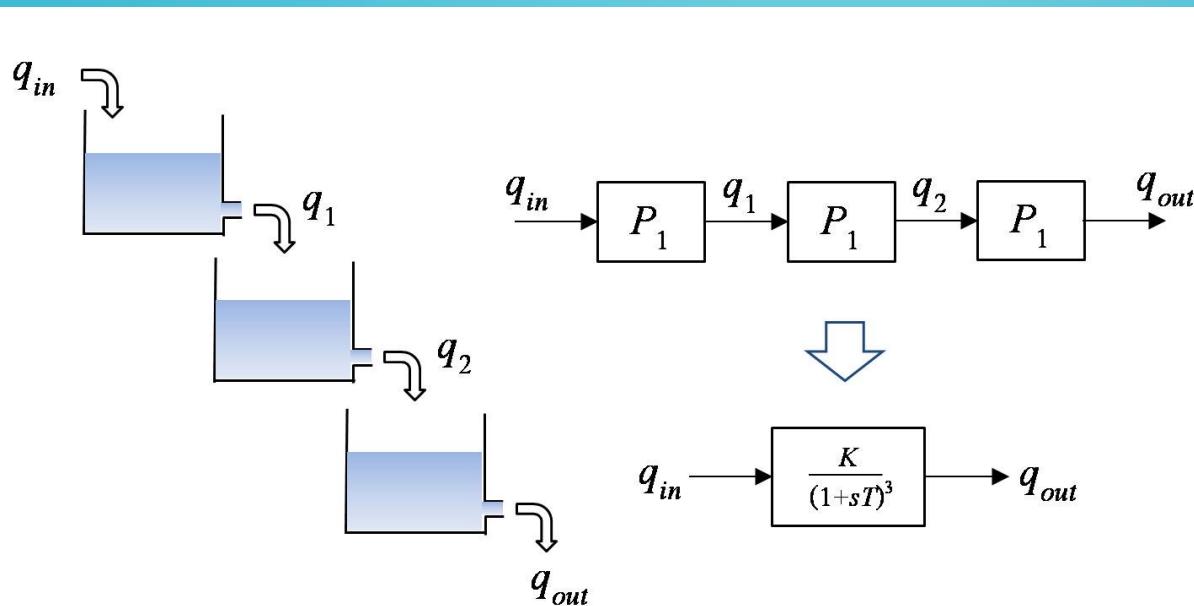
เตรียมพร้อมก่อนทำ EXERCISE 1

- ศึกษาไฟล์โปรแกรม **netpie_cmdtest.py**
- ศึกษาวิดีโอ [ส่งคำสั่งจาก NETPIE2020 ไปยังโปรแกรม micropython บน ESP32](#) จากหน้า **micropython github**

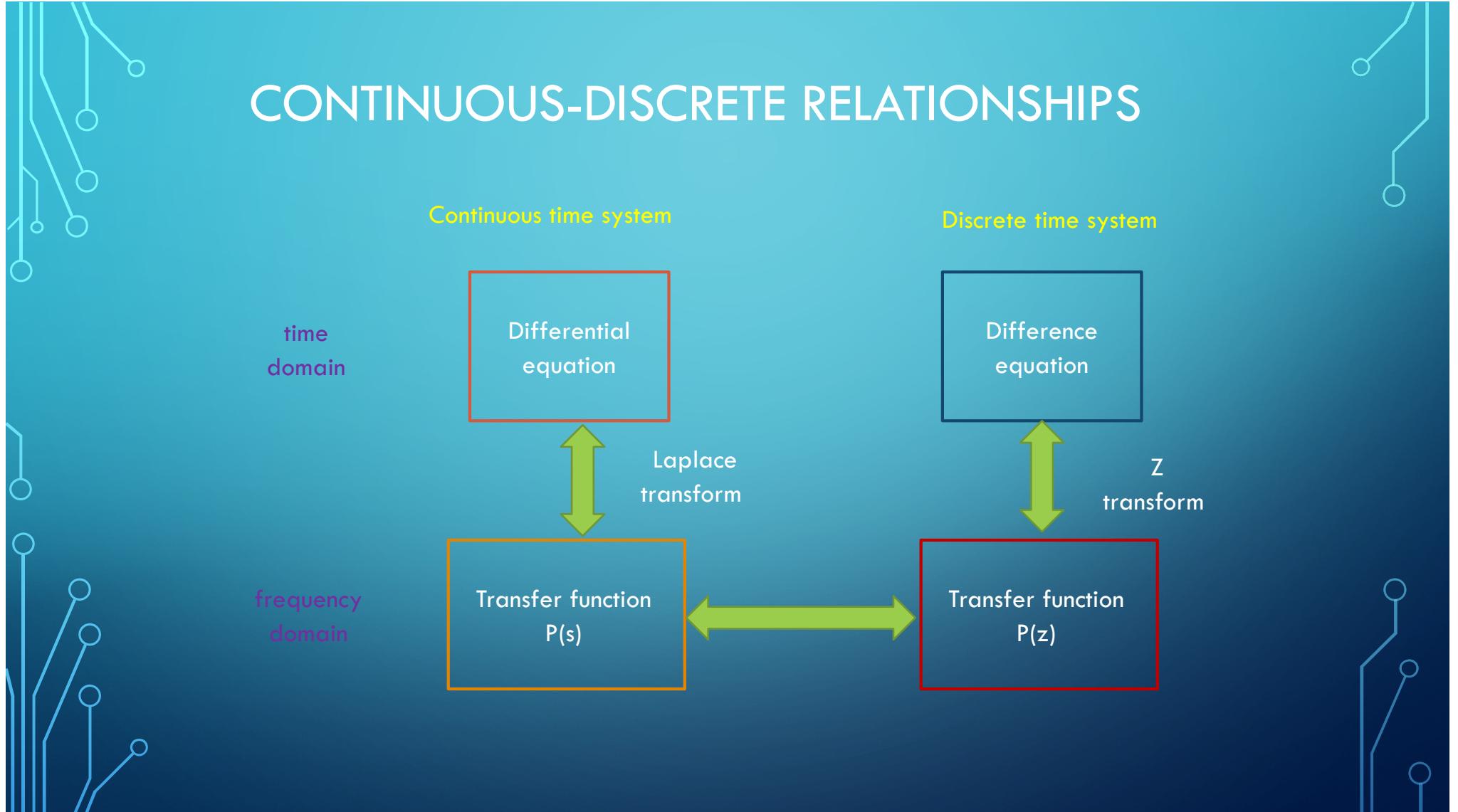
EXERCISE 1 : ใช้เวลาประมาณ 30 นาที

- พัฒนาโปรแกรมโดย micropython เพื่อสั่งงาน 3 คำสั่งกับ on-board LED
 - led=0/1 : สั่งให้ LED ดับ หรือติดสว่าง ด้วยค่าความสว่างที่กำหนดโดยคำสั่ง brightness
 - toggle=0/1 : สั่งให้ LED กระพริบตามความเวลา 200 ms
 - brightness=x : ปรับค่าความสว่างของ LED จาก 0 (ดับ) ถึง 1 (สว่างสุด)
- ใช้เค้าโครงโปรแกรม ex1_netpie_cmptest.py โดยเติมโค้ดส่วนที่ขาดให้สมบูรณ์
- สร้าง Freeboard เพื่อสั่งงานกับ ESP32 (สามารถ import ไฟล์ dashboard.json)

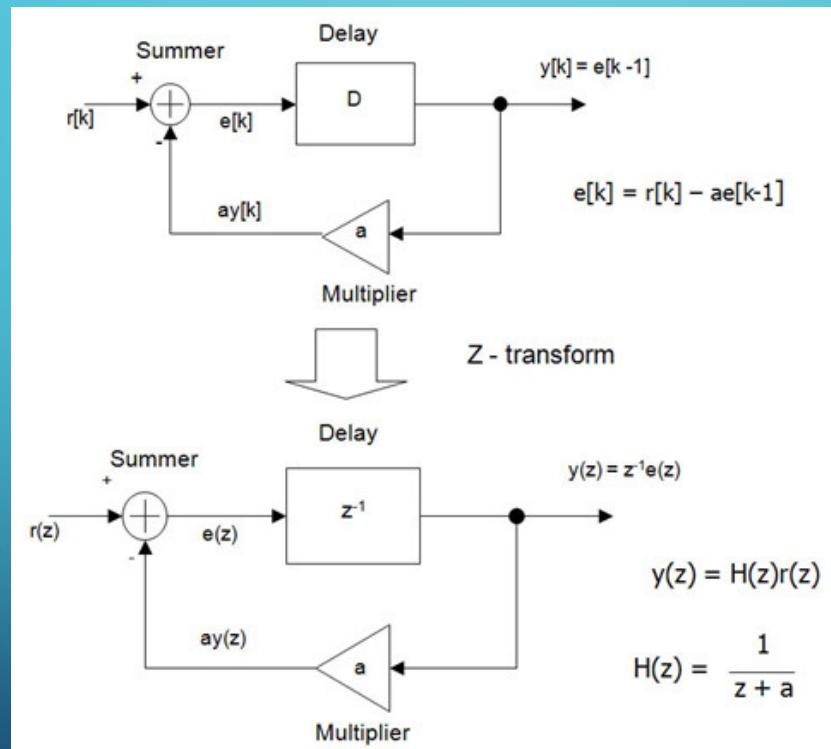
การจำลองพลวตระบบเชิงเส้น โดยคอมพิวเตอร์



CONTINUOUS-DISCRETE RELATIONSHIPS



DISCRETE-TIME SYSTEM DESCRIPTION



Difference equation

Discrete-time
Transfer function

TRANSFORMATION OF C(S) TO C(Z)

Forward difference

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

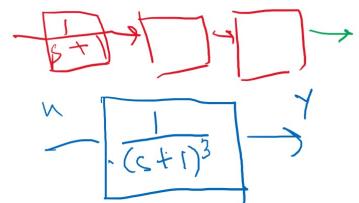
Backward difference

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

Bilinear transform

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

แปลง $1/(s+1)$ เป็นระบบดิจิทัลรีตและเป็นสมการผลต่าง



Bilinear Transform

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

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



$$\frac{y_1(z)}{u(z)} = \frac{1 + z^{-1}}{a + bz^{-1}}$$

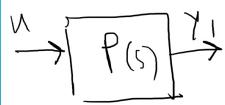
$$\frac{(z-1)Tz + T}{(z-1)(z+T)z + T-2} \checkmark = \frac{T + Tz^{-1}}{(z+T) + (T-z)z^{-1}}$$

$$y_1(z) = (T + Tz^{-1})u(z)$$

$$\Downarrow ay_1(k) + by_1(k-1) = T(u(k)) + T(u(k-1))$$

จำลองเบื้องต้นโดยเขียนโค้ดบน JUPYTER NOTEBOOK (1)

$$P(s) = \frac{1}{s+1} \rightarrow P(z) = \frac{T+z^{-1}}{a+bz^{-1}}$$



$$ay_1(k) + by_1(k-1) = T(u(k) + u(k-1))$$

$$y_1(k) = \frac{1}{a} (-by_1(k-1) + T(u(k) + u(k-1)))$$

```
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 input 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)

$$P(z) = \frac{1}{S+1} \Rightarrow P(z) = \frac{T+z^{-1}}{a+bz^{-1}}$$

$$a = c_2 + T$$

$$b = (T - z)$$

$\xrightarrow{\text{input}} [P(z)] \xrightarrow{\gamma_1}$

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

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

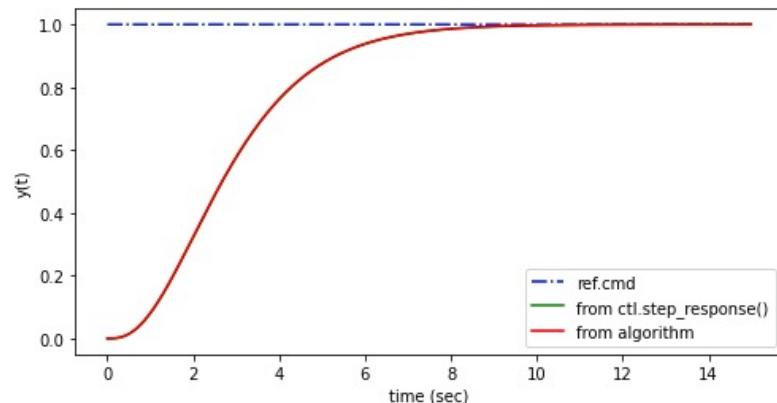
```
def lag3(a,b,T, u, u_states, y_states):
    for k in range(3):
        y_states[2*k] = y_states[2*k+1]
        u_states[2*k] = u_states[2*k+1]
        if k == 0:
            u_states[2*k+1] = u
        else:
            u_states[2*k+1] = y_states[2*k-1]
        y_states[2*k+1] = (1/a)*(-b*y_states[2*k]+T*(u_states[2*k+1]+u_states[2*k]))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

```
# 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
```

จำลองเบื้องต้นโดยเขียนโค้ดบน JUPYTER NOTEBOOK (3)

In [31]:

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



จำลองบน ESP32 (1)

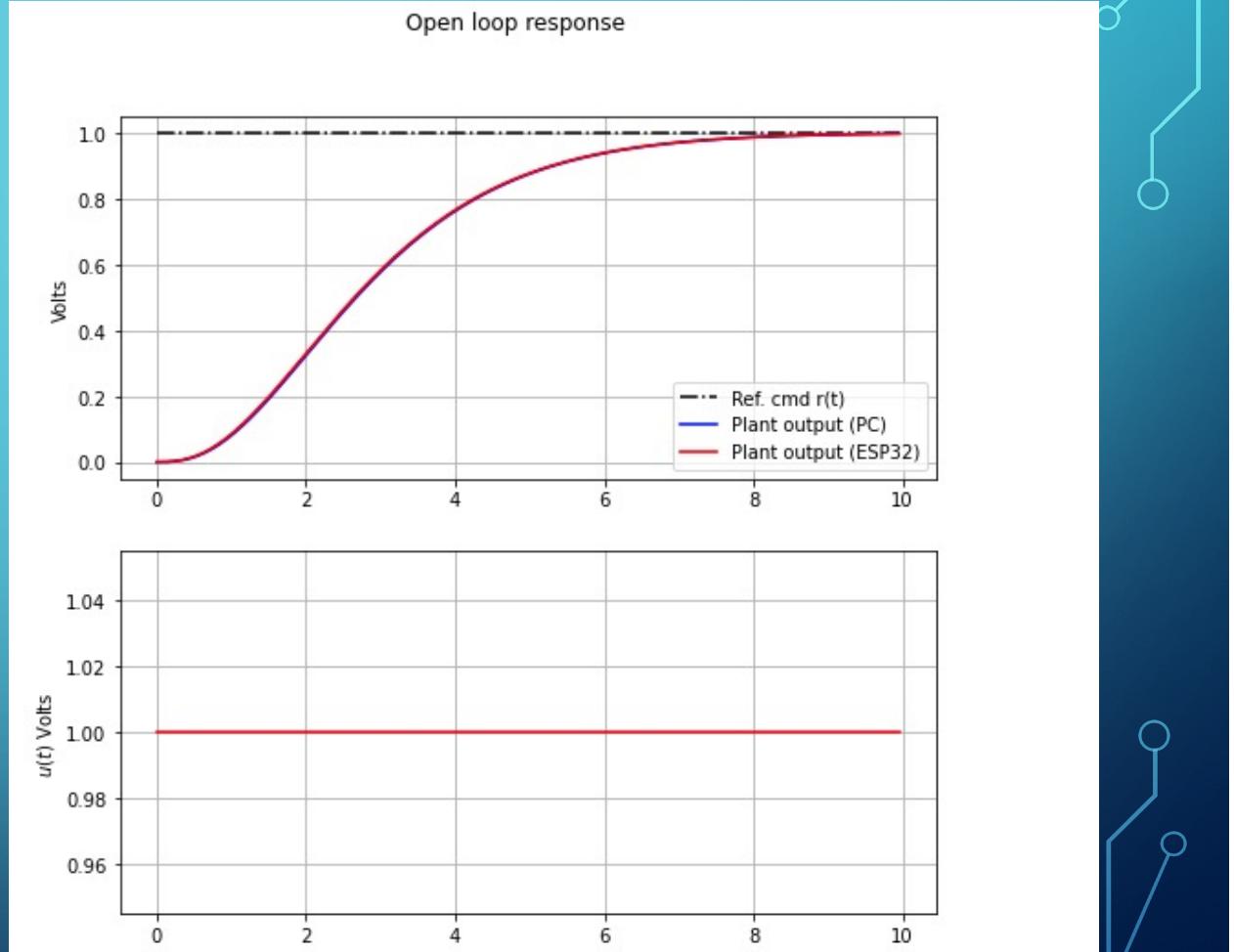
```
lag3.plantsim.py %
23 u = 1
24
25 def lag3(a,b,T, u, u_states, y_states):
26     for k in range(3):
27         y_states[2*k] = y_states[2*k+1]
28         u_states[2*k] = u_states[2*k+1]
29         if k == 0:
30             u_states[2*k+1] = u
31         else:
32             u_states[2*k+1] = y_states[2*k-1]
33             y_states[2*k+1] = (1/a)*(- b*y_states[2*k]+T*(u_states[2*k+1]+u_states[2*k]))
34     return y_states[5]
35
36
37 for i in range(datasize):
38     if i==0:
39         print("datamat = np.array([]")
40
41     while dt < T: # block execution until T expires
42         t_current = time.ticks_ms()
43         dt = t_current - t_previous
44
45         y = lag3(a,b,T,u, u_states, y_states)
46         print("{}, {}, {}, {}, ".format(t,u,y,u))
47         if i==datasize-1:
48             print("])")
49         t+=T
50

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

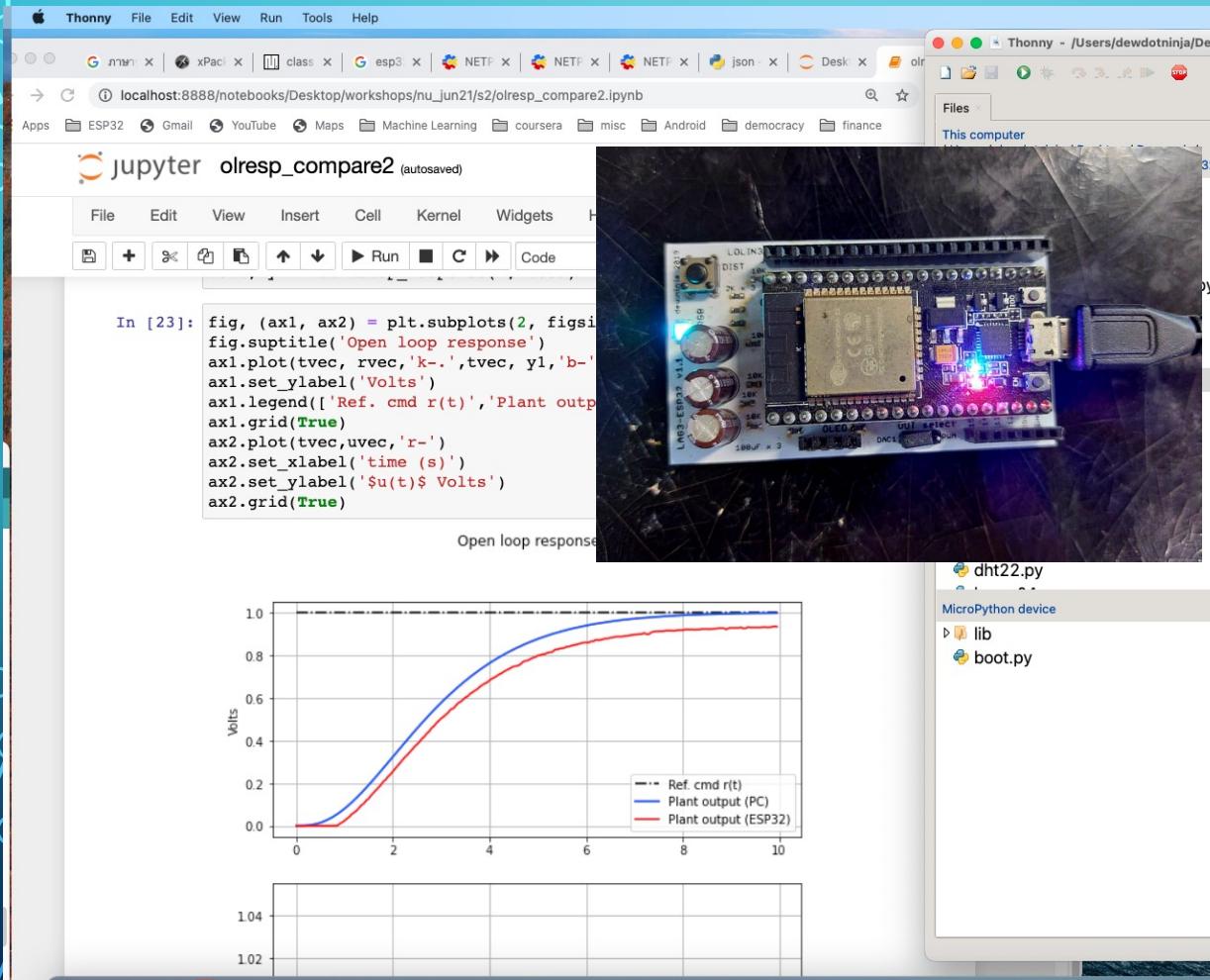
datamat = np.array([
[0,1,1.450937e-05,1],
[0.05,1,9.944225e-05,1],
[0.1,1,0.0003459548,1],
[0.15,1,0.0008456681,1],
[0.2,1,0.001676403,1],
[0.25,1,0.002901642,1],
[0.3,1,0.004573871,1],
[0.35,1,0.006734811,1],
[0.4,1,0.009416744,1],
[0.45,1,0.01264355,1],
[0.5,1,0.01643163,1],
```

จำลองบน ESP32 (2)

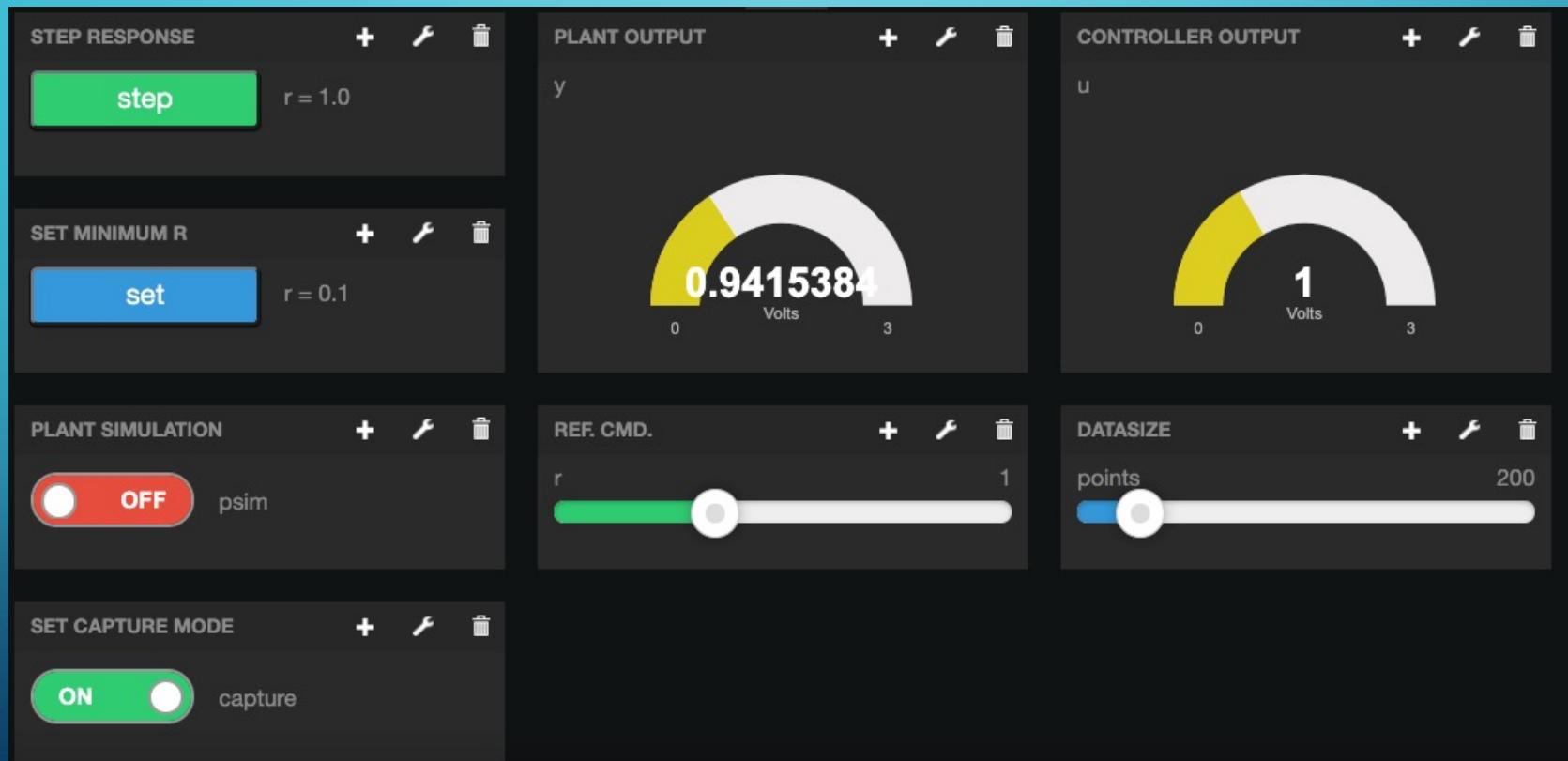
เปรียบเทียบผลตอบสนอง
ระหว่างการจำลองบน PC
(Python control library) กับ
การจำลองบน ESP32



เปรียบเทียบผลตอบสนองของเปิดของ LAG3



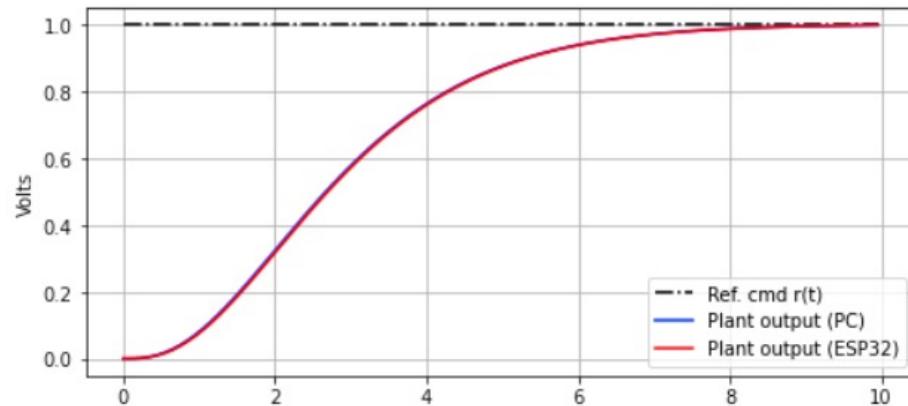
NETPIE FREEBOARD สำหรับ LAG3



ผลตอบสนองวงเปิดจากการจำลองโดย อัลกอริทึม

```
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)
```

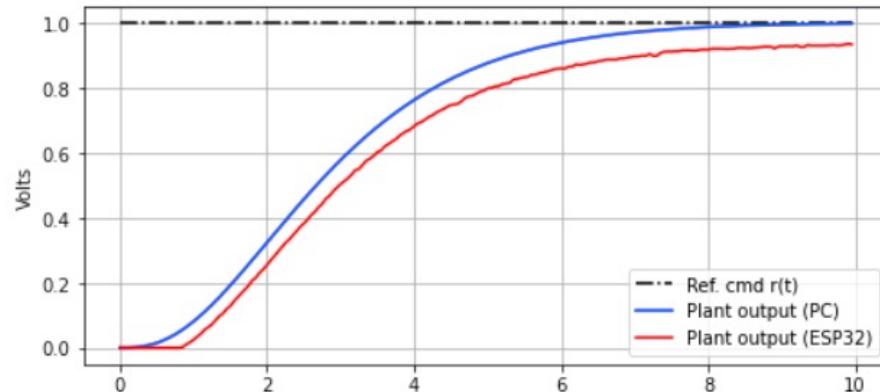
Open loop response



ผลตอบสนองวุ่งเปิดจาก การจำลองโดย用จาร อิเล็กทรอนิกส์

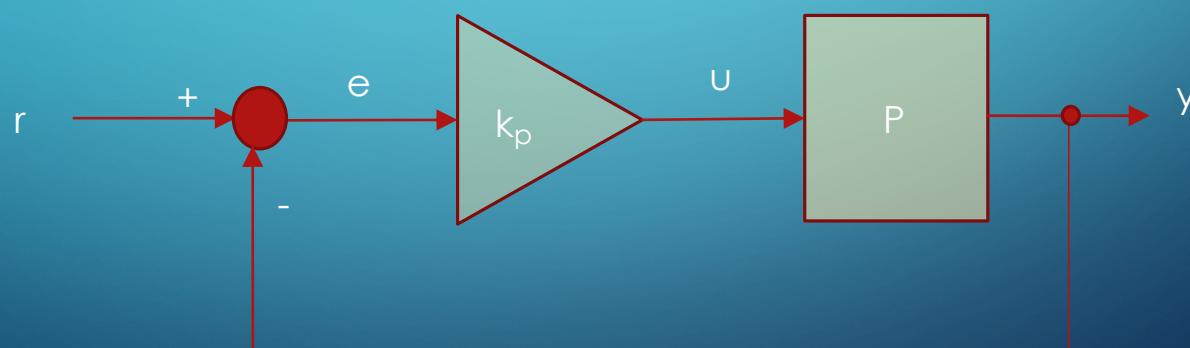
```
In [23]: fig, (ax1, ax2) = plt.subplots(2, figsize=(8,8))
fig.suptitle('Open loop response')
ax1.plot(tvec, rvec,'k-.',tvec, y1,'b-',tvec,yvec,'r-')
ax1.set_ylabel('Volts')
ax1.legend(['Ref. cmd r(t)', 'Plant output (PC)', 'Plant output (ESP32)'])
ax1.grid(True)
ax2.plot(tvec,uvec,'r-')
ax2.set_xlabel('time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

Open loop response



EXERCISE 2 : ใช้เวลาประมาณ 30 นาที

- เพิ่มตัวควบคุม proportional control
- เพิ่ม widget บน Freeboard ดังนี้
 - ปรับค่าพารามิเตอร์ K_p
 - Toggle switch สำหรับสลับค่าระหว่างระบบบางเปิดกับระบบบางปิด



Q&A SESSION

Thank You

