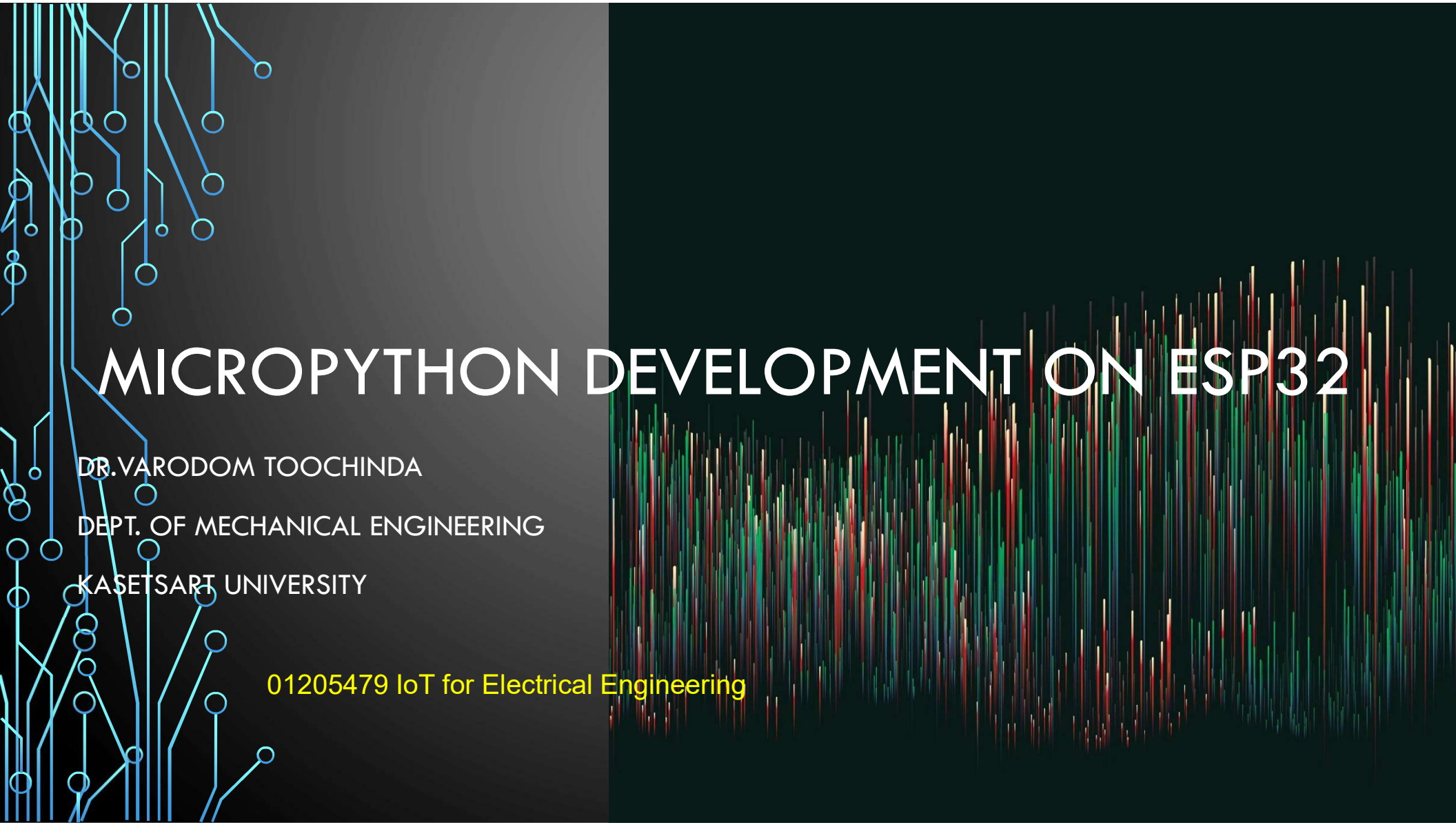


MICROPYTHON DEVELOPMENT ON ESP32

DR.VARODOM TOOCHINDA

DEPT. OF MECHANICAL ENGINEERING

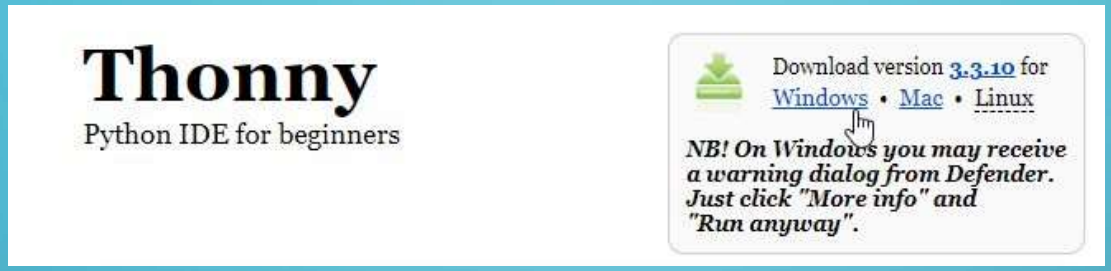
KASETSART UNIVERSITY

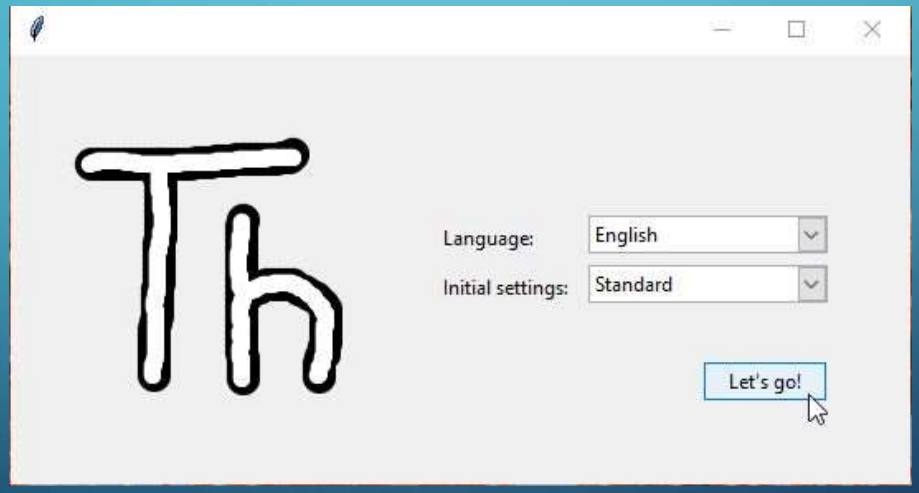
01205479 IoT for Electrical Engineering

TOPICS

- Thonny IDE for micropython development on ESP32
- Basic Python programming for ESP32
- LTI simulation ESP32
- Controller implementation on micropython

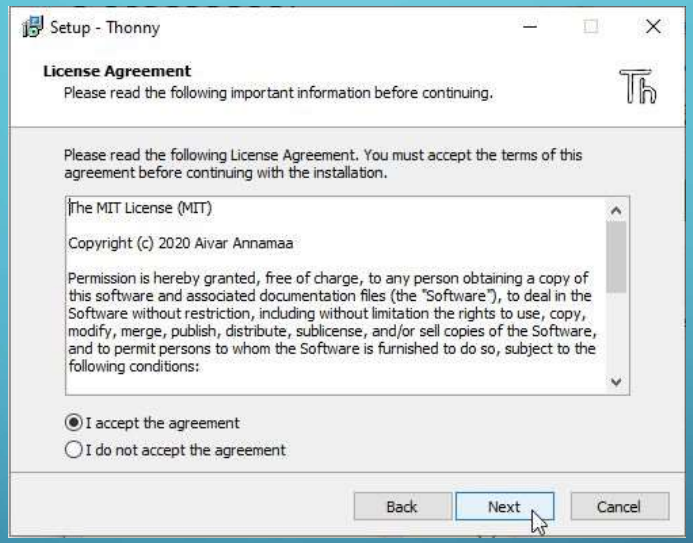
DOWNLOAD AND INSTALL THONNY IDE

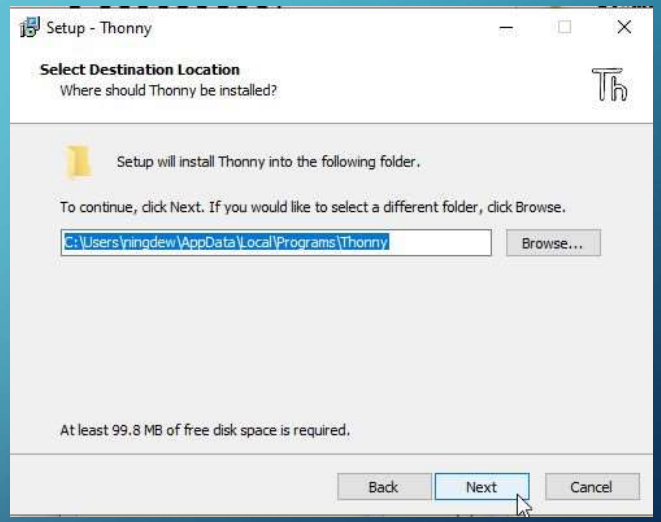




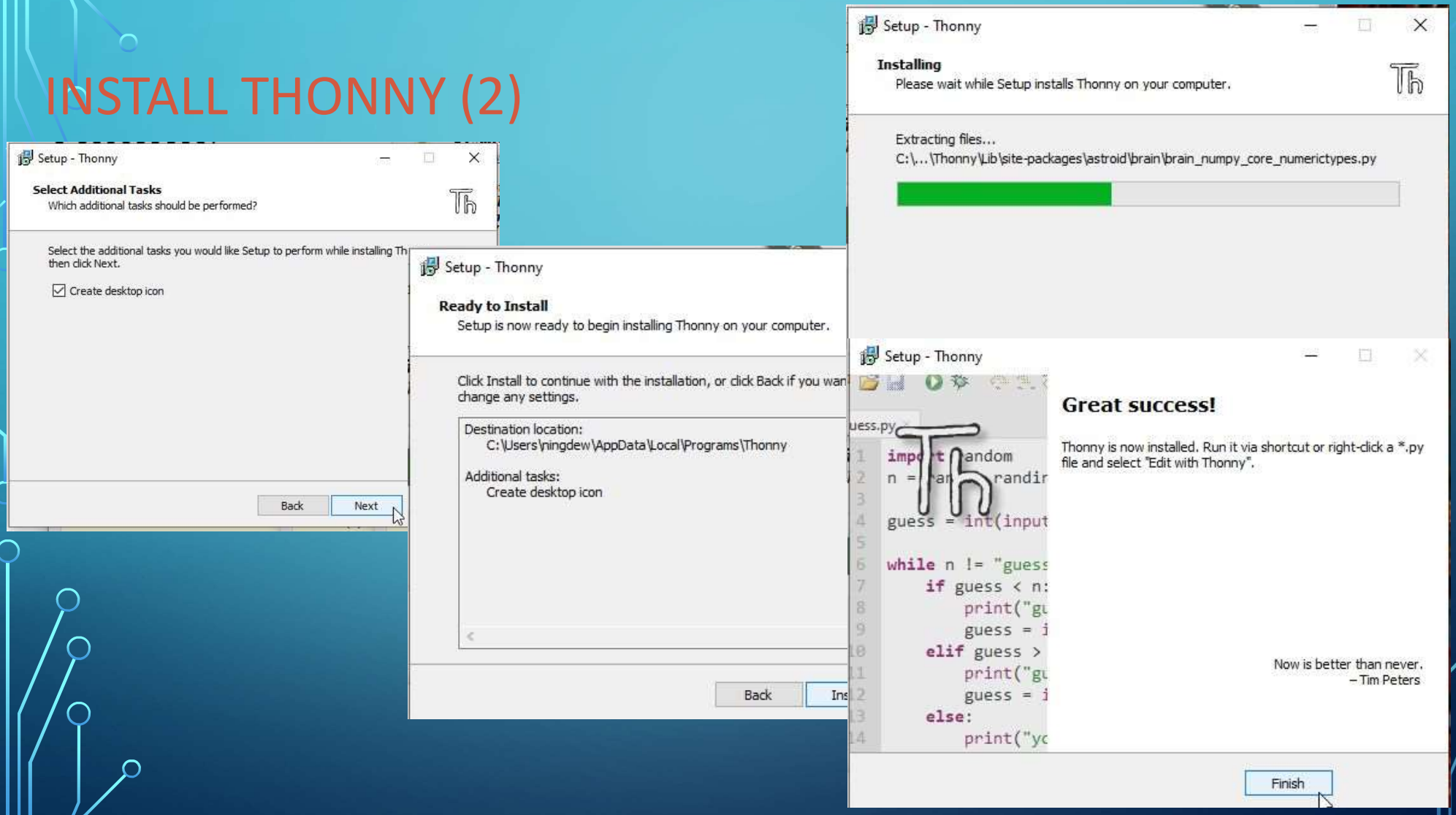
Note : newer version may have slightly different look, but the steps are the same.

INSTALL THONNY (1)

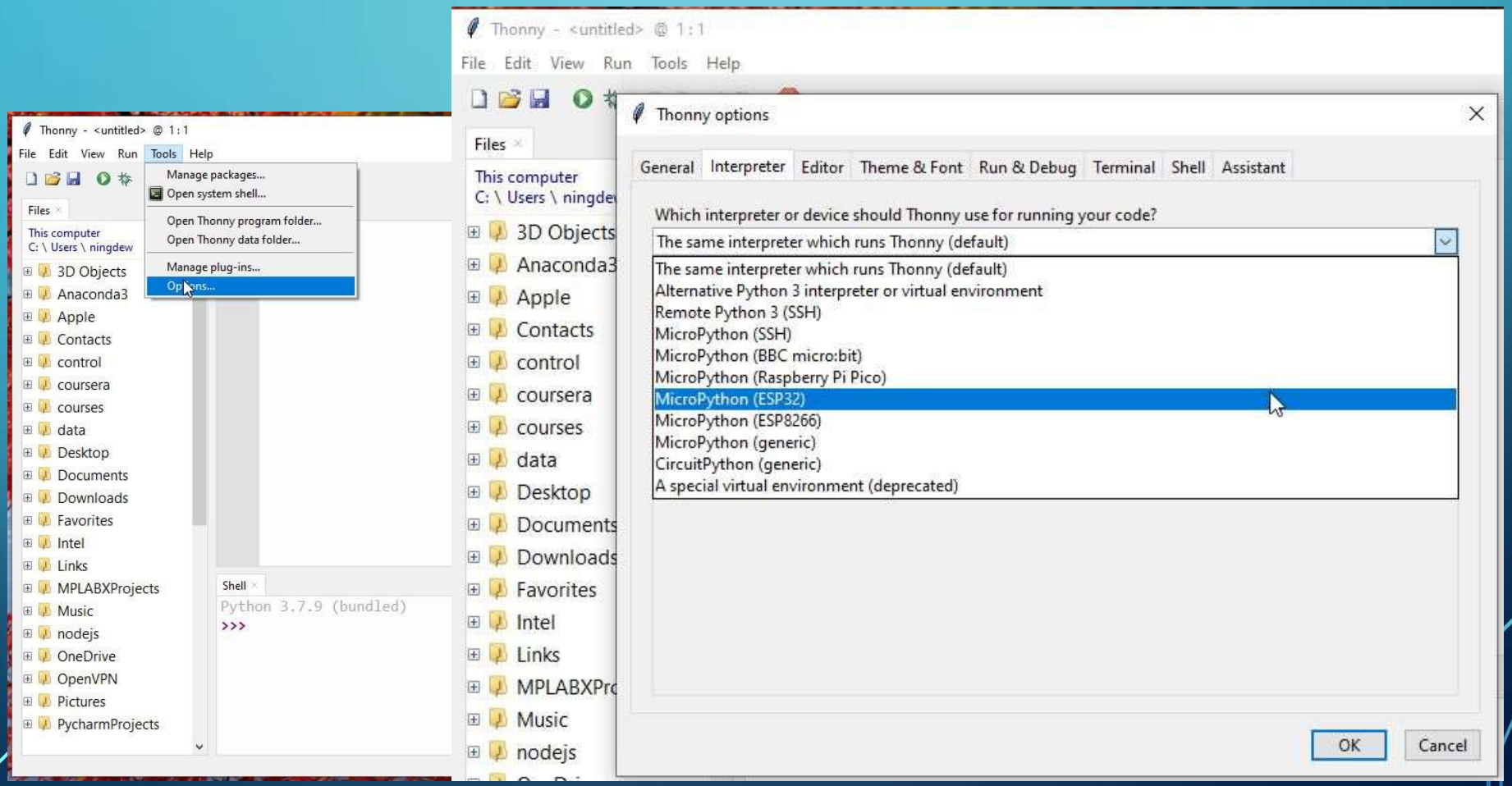




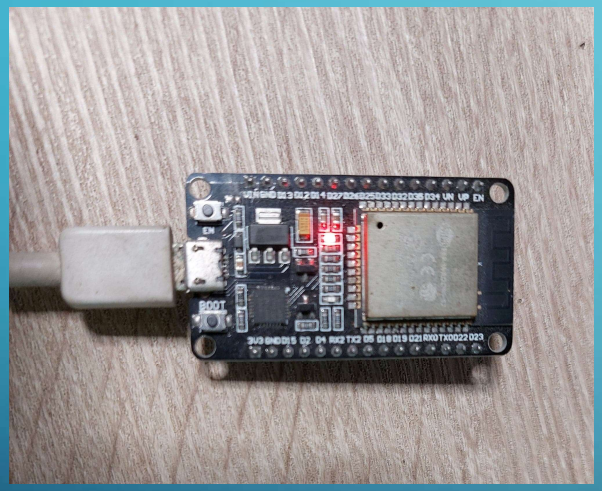
INSTALL THONNY (2)

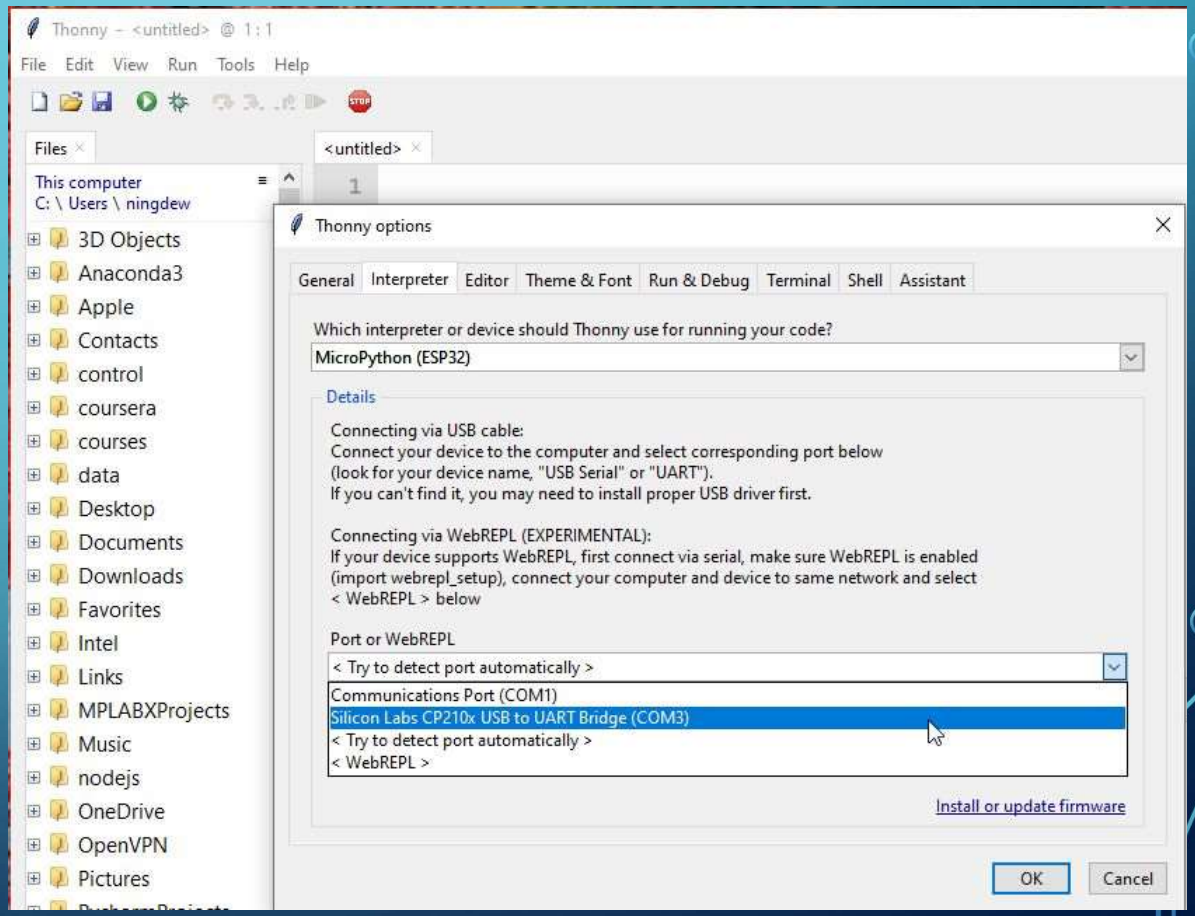


OPEN THONNY AND CHOOSE OPTION →INTERPRETER

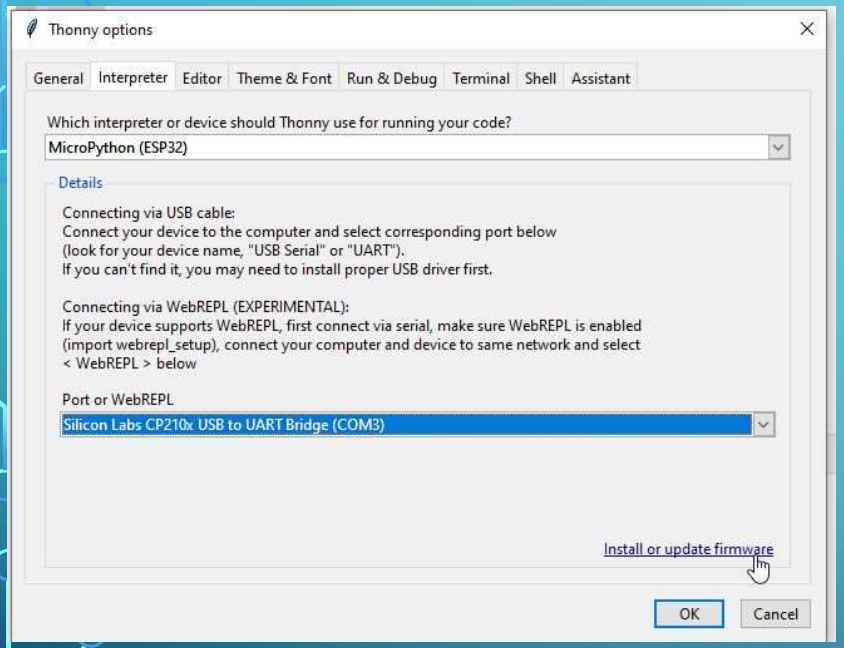


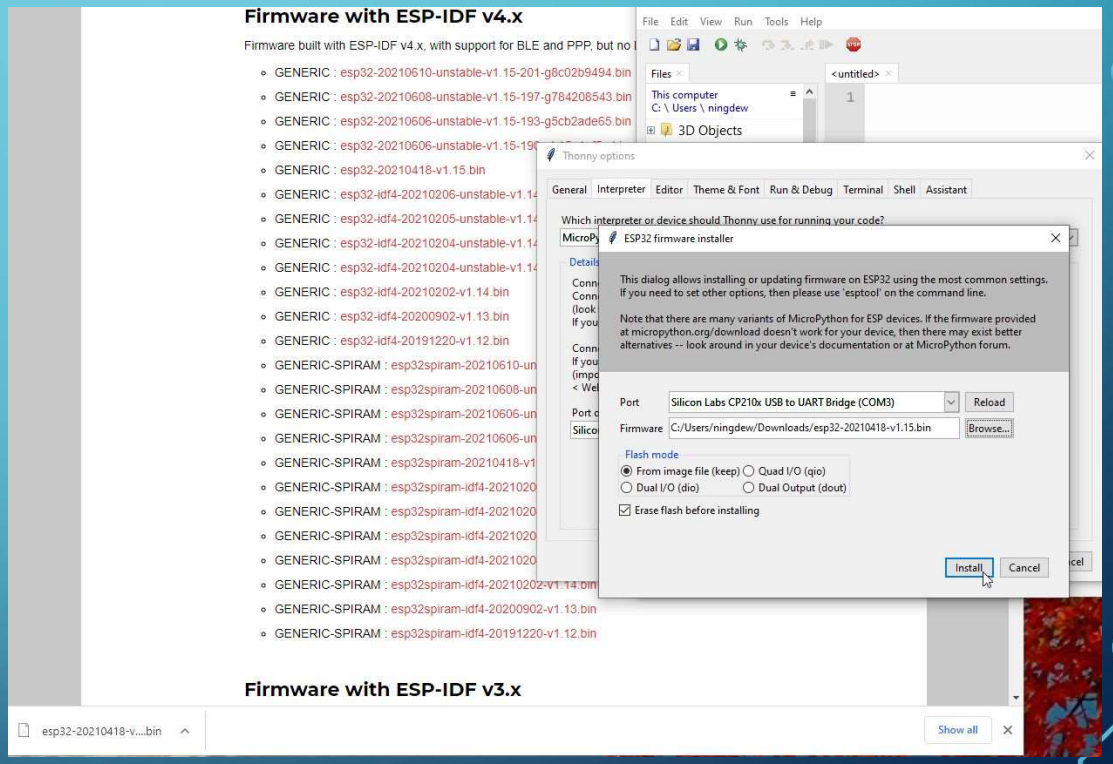
CONNECT ESP32 TO USB AND INSTALL FIRMWARE (1)



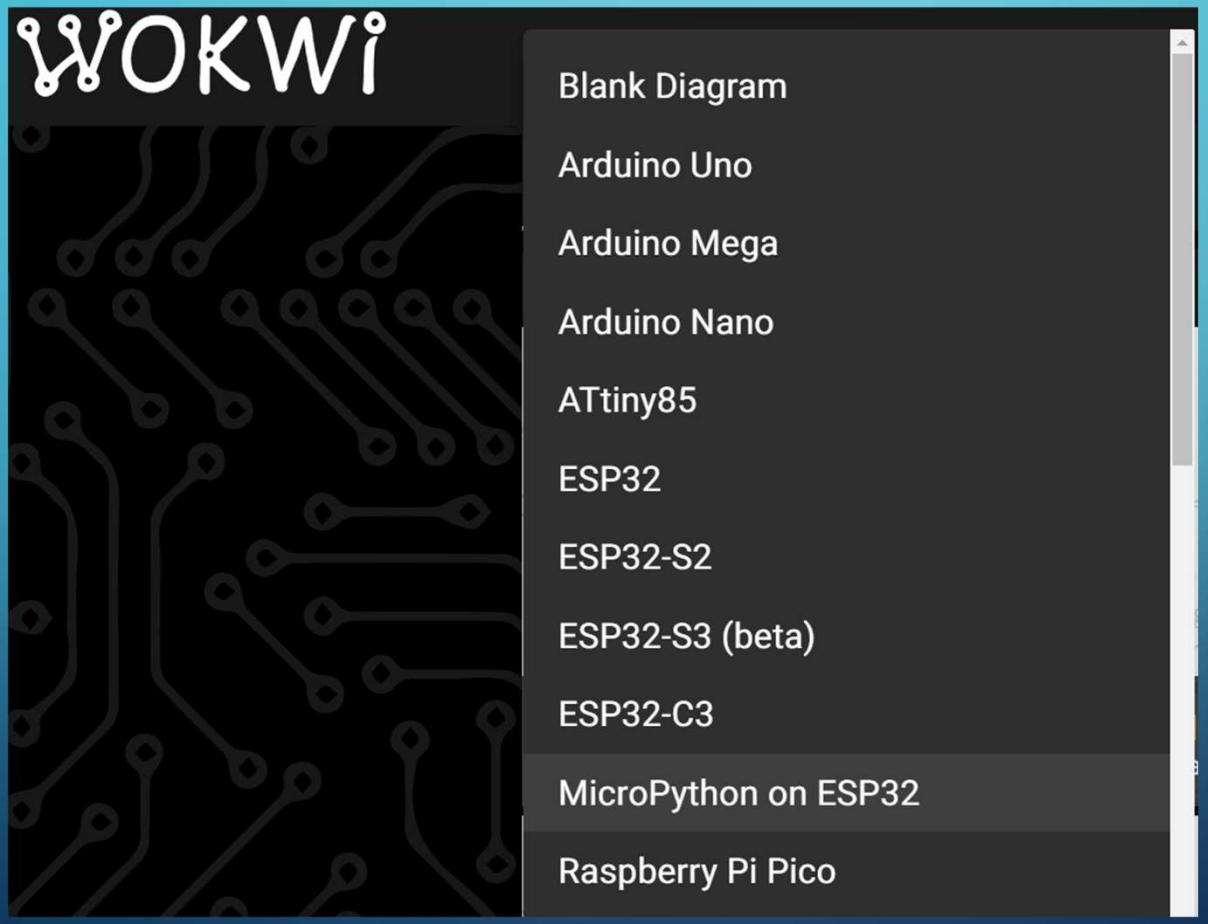


INSTALL FIRMWARE (2)

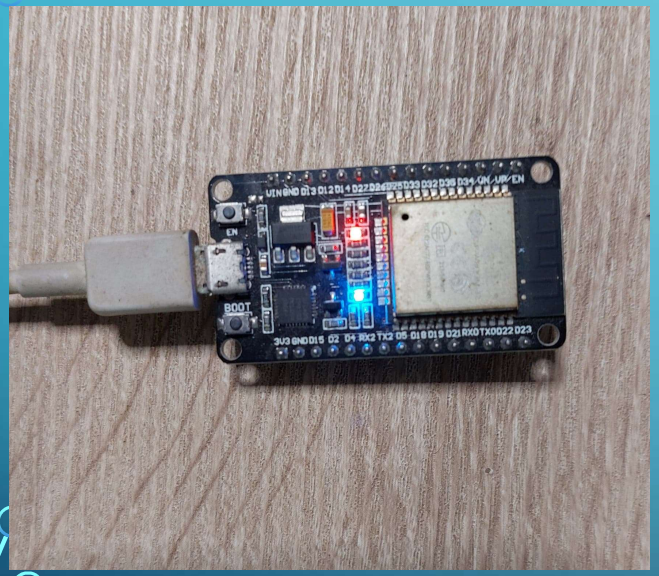




Simulation on Wokwi

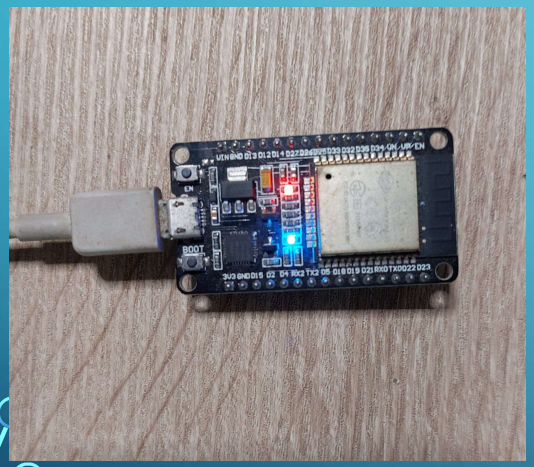


LED_BLINK.PY



```
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)
```

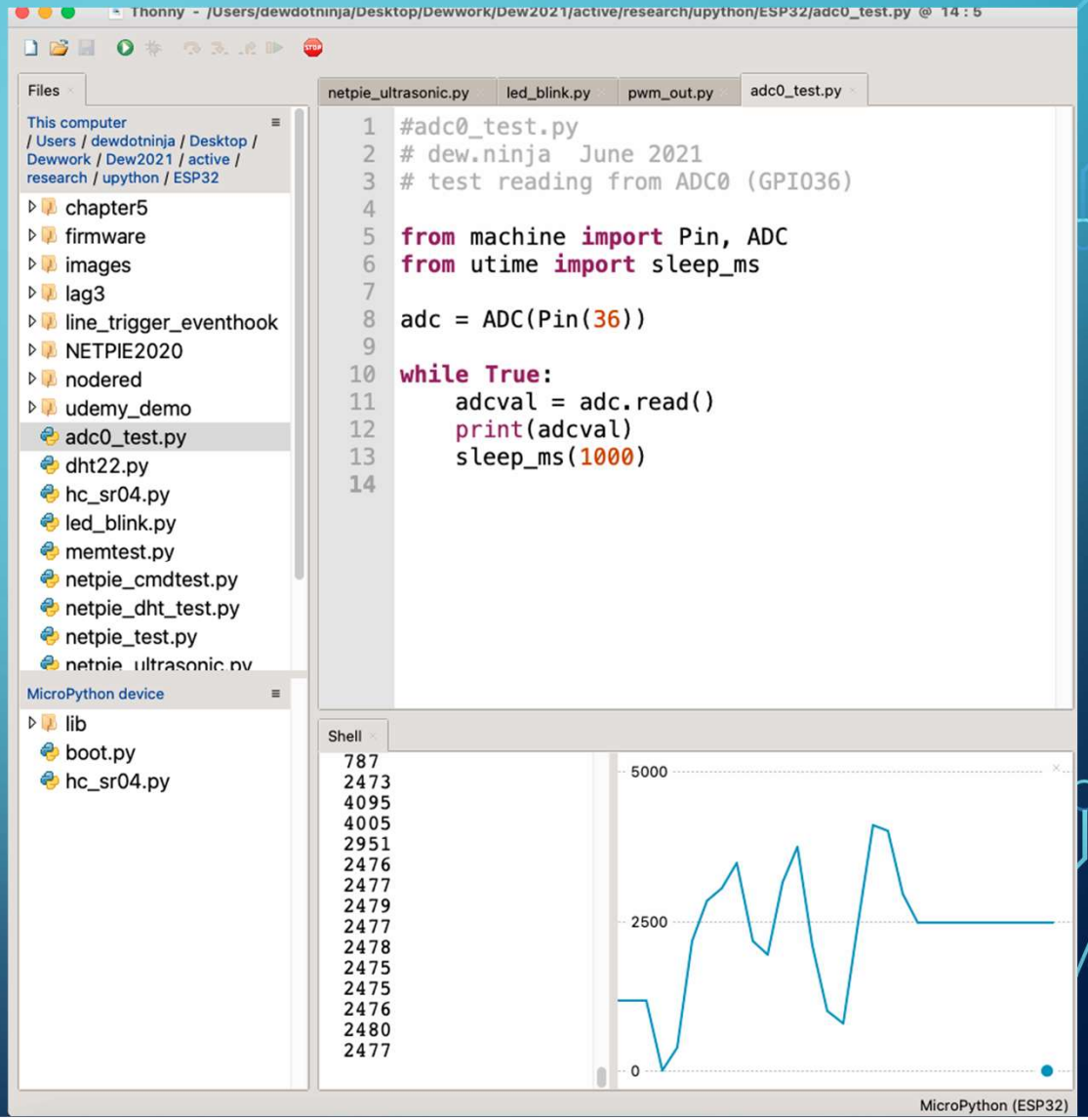
TEST PWM (PWM_OUT.PY)



```
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)
```

TEST ADC (ADC0_TEST.PY)

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



Read digital input

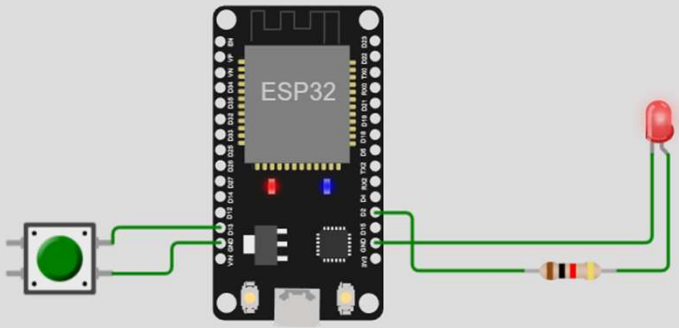
din_test.py

main.py ● diagram.json ●

```
1   # din_test.py
2   # dew.ninja June 2023
3   # test digital input with switch on NodeMCU-32S
4
5   import math
6   import time
7   from time import sleep_ms
8   from machine import Pin, PWM
9
10  obled = PWM(Pin(2))  # on-board led
11  button = Pin(13,Pin.IN, Pin.PULL_UP) # pushbutton switch
12
13  PWMMAX = 1023
14  PWMMID = 511
15  PWMMIN = 0
16
17  # genchirp() check button status
18  # and generate chirp signal if pressed
19
20  def genchirp():  # check button status and generate chirp signal if pr
21      stepnum = 10
22      sinstep = 6.28/stepnum
23      sinangle = 0
24      dt = 30
25      cyclecomplete = False
26      if not button.value():
27          sleep_ms(100)  # debounce
28          while not button.value(): # blocked if switch pressed
```

Simulation

02:21.005




```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp
:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4656
load:0x40078000,len:13284
ho 0 tail 12 room 4
load:0x40080400,len:3712
entry 0x4008064c
```

DAC1_TEST.PY

Note : DAC does not work on Wokwi simulation!

```
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))
while True:
    adcval = adc.read()
    dacval = int(adcval/4)
    if dacval>DACMAX:
        dacval = DACMAX
    dac1.write(dacval)
    sleep_ms(1000)
```

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()))
```

Utilize timer by writing callback function (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
```

Set period using time library

time_ticks.py

```
import time
from machine import Pin
led= Pin(2, Pin.OUT)          # GPIO2 as led output
blink_period = 1000
time_prev = 0
while True:
    time_current = time.ticks_ms()
    if time_current<time_prev:
        time_prev = time_current # prevent overflow
    if (time_current - time_prev)>blink_period:
        led.value(not led.value())  # blink on-board led
        time_prev = time_current
```

DHT SENSOR (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)
```

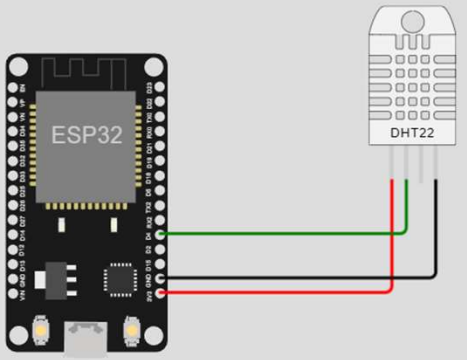
DHT22 simulation on Wokwi

WOKWI SAVE SHARE Do

main.py ● diagram.json ●

```
1  #DHT22_wokwi.py
2  # dew.ninja  Sep 2023
3
4  import dht
5  import machine
6  from utime import sleep_ms
7  d = dht.DHT22(machine.Pin(4))
8  while True:
9
10     d.measure()
11     temperature=d.temperature()
12     humidity=d.humidity()
13     print("temperature = ",temperature)
14     print("humidity = ",humidity)
15     sleep_ms(2000)
16
```

Simulation




```
temperature =  24.0
humidity =  40.0
temperature =  24.0
humidity =  40.0
temperature =  24.0
humidity =  40.0
temperature =  24.0
humidity =  40.0
temperature =  24.0
humidity =  40.0
```

DHT TEST (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
```

cmd_interpreter.py (or cmd_interpreter_ww.py for wokwi)

main.py ● diagram.json

```
1   # cmd_interpreter_ww.py
2   # dew.ninja  August 2023
3   # for Wokwi simulation. Use input() instead of sys.stdin.readline()
4   # Seperate command from parameter
5
6   #import sys
7   from utime import sleep_ms
8
9   #command interpreter function
10  def cmdInt(userstr):
11      result = userstr.find("=")
12      if result == -1:
13          noparm = 1
14          cmdstr = userstr.strip()
15      else:
16          noparm = 0
17          splitstr = userstr.split("=")
18          cmdstr = splitstr[0].strip()
19          parmstr = splitstr[1].strip()
20      print("command = " +cmdstr)
21      if noparm:
22          print('No parameter in command')
23      else:
24          print("parameter = " + parmstr)
25
26  def user_input():
27      print('====================')
```

Simulation

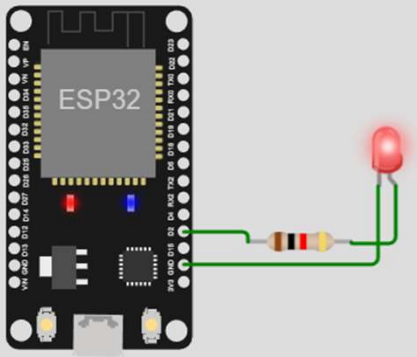
```
led=1
command = led
parameter = 1
====================
Enter command :
dbupdate
command = dbupdate
No parameter in command
====================
Enter command :
```

cmdtest2_ww.py

main.py diagram.json

```
1   # cmdtest2_ww.py
2   # dew.ninja Sep 2023
3   # for Wokwi simulation
4
5   from machine import Pin, Timer
6   #import sys
7   from utime import sleep_ms
8
9   led = Pin(2, Pin.OUT)
10  led_period = 1000  # default
11  toggle_mode = 1
12  led_status = 0
13
14  timer=Timer(1)
15  def timer_isr(event):
16      led.value(not led.value())
17
18  timer.init(period=led_period, mode=Timer.PERIODIC,
19                                callback=timer_isr)
20
21  # helper function for cmdInt()
22  # separate command and parameter
23  # at "="
24  def splitCmd(userstr):
25      result = userstr.find("=")
26      if result == -1:
27          noparm = 1
28          cmdstr = userstr.strip()
```

Simulation




```
Enter command :
toggle=0

Enter command :
led
Current led status = 1

Enter command :
period
Current led period = 1000

Enter command :
```

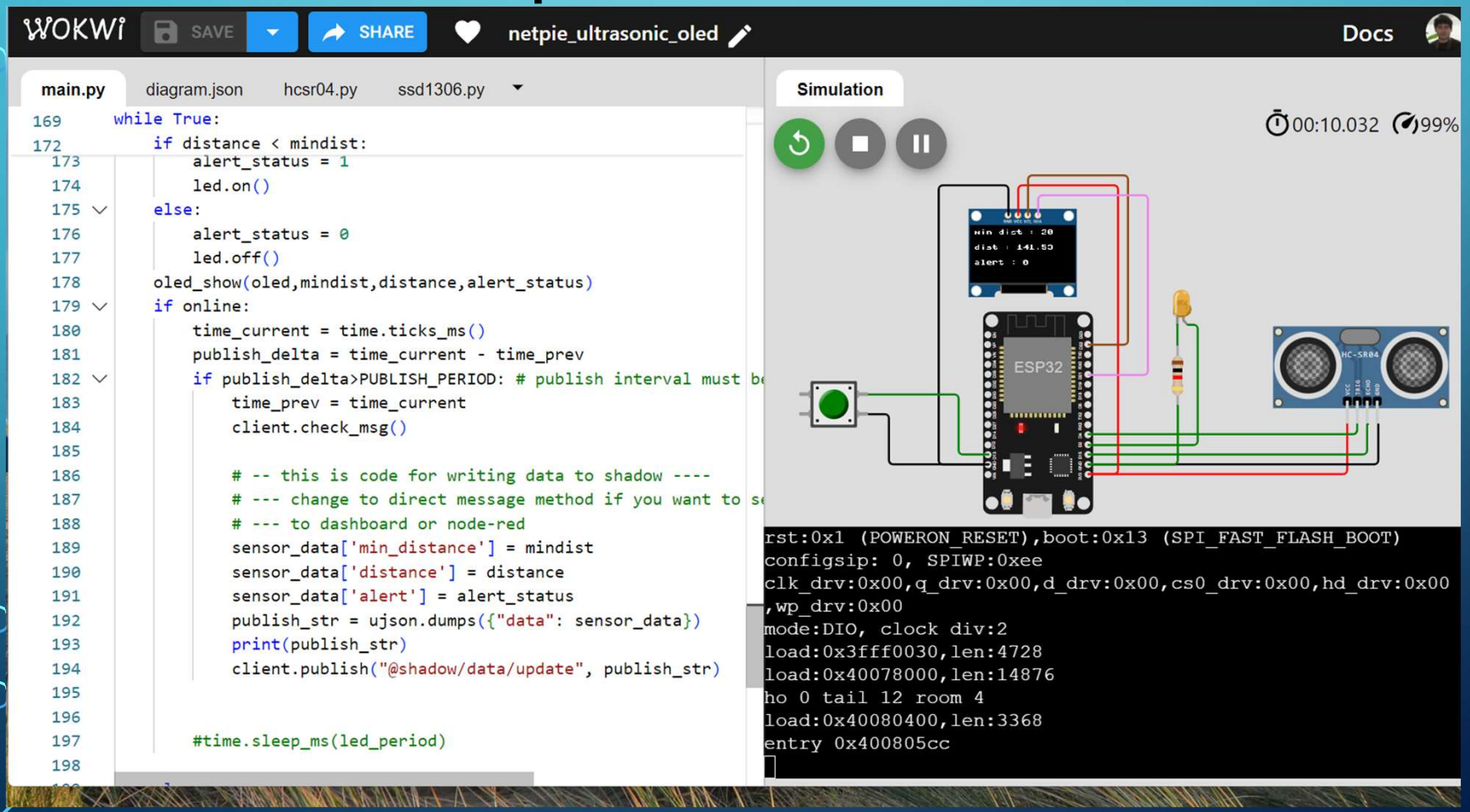
Exercise 1:

- Develop a program to change RGB LED color on LAG3 board randomly (or on Wokwi)
- R, G, B are connected to pin 19,18,17, respectively. (on Wokwi, change pins as you like.)
- Rate to change color is adjustable via command rate = x (Hz)
- The command interpreter must limit rate to within 0.1 – 10 Hz

Template on wokwi : https://wokwi.com/projects/406465660086615041

HCSR04 example

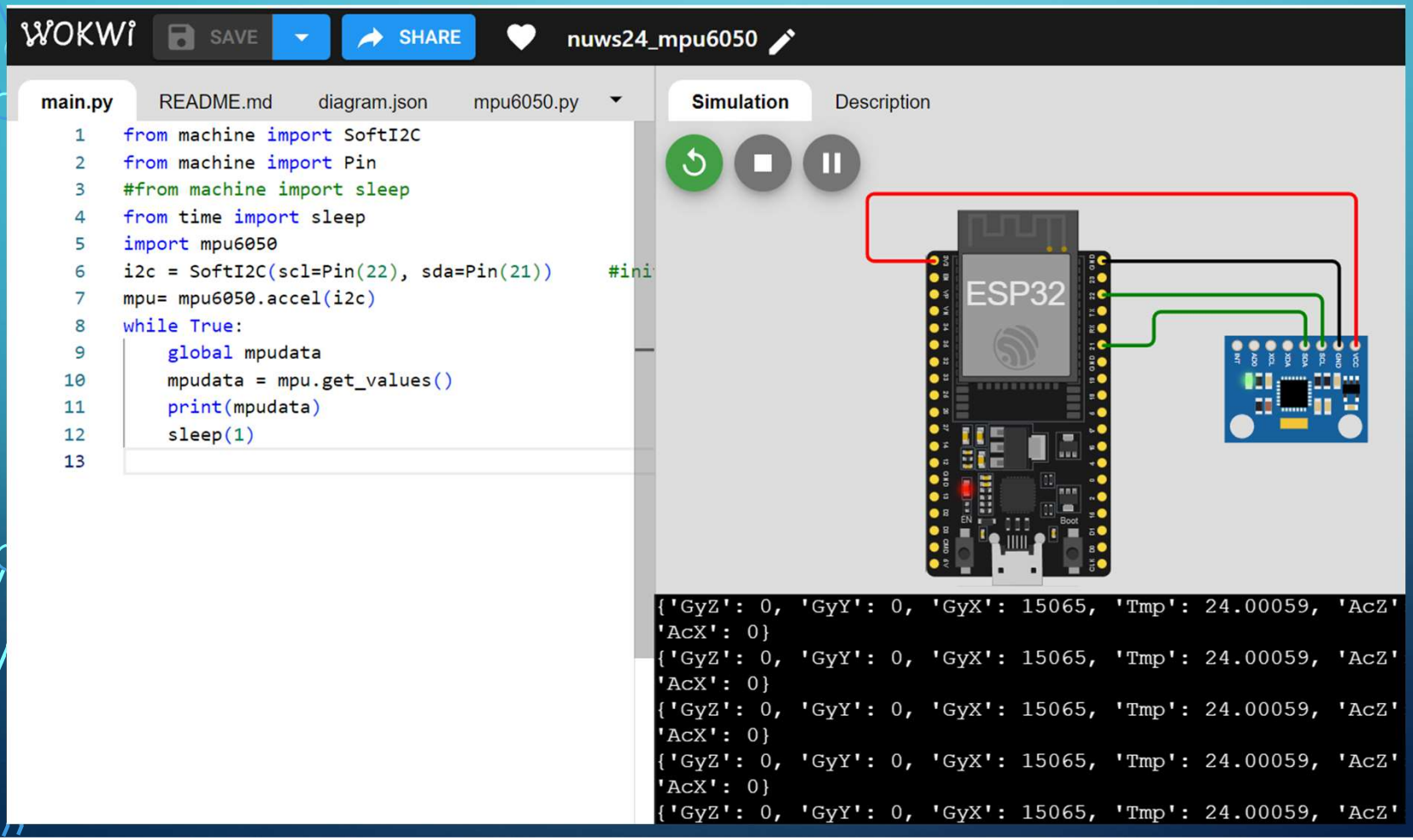
https://wokwi.com/projects/406627227063083009




```
169 while True:
172     if distance < mindist:
173         alert_status = 1
174         led.on()
175     else:
176         alert_status = 0
177         led.off()
178     oled_show(oled,mindist,distance,alert_status)
179     if online:
180         time_current = time.ticks_ms()
181         publish_delta = time_current - time_prev
182         if publish_delta>PUBLISH_PERIOD: # publish interval must be
183             time_prev = time_current
184             client.check_msg()
185
186             # -- this is code for writing data to shadow ----
187             # --- change to direct message method if you want to se
188             # --- to dashboard or node-red
189             sensor_data['min_distance'] = mindist
190             sensor_data['distance'] = distance
191             sensor_data['alert'] = alert_status
192             publish_str = ujson.dumps({"data": sensor_data})
193             print(publish_str)
194             client.publish("@shadow/data/update", publish_str)
195
196
197         #time.sleep_ms(led_period)
198
```

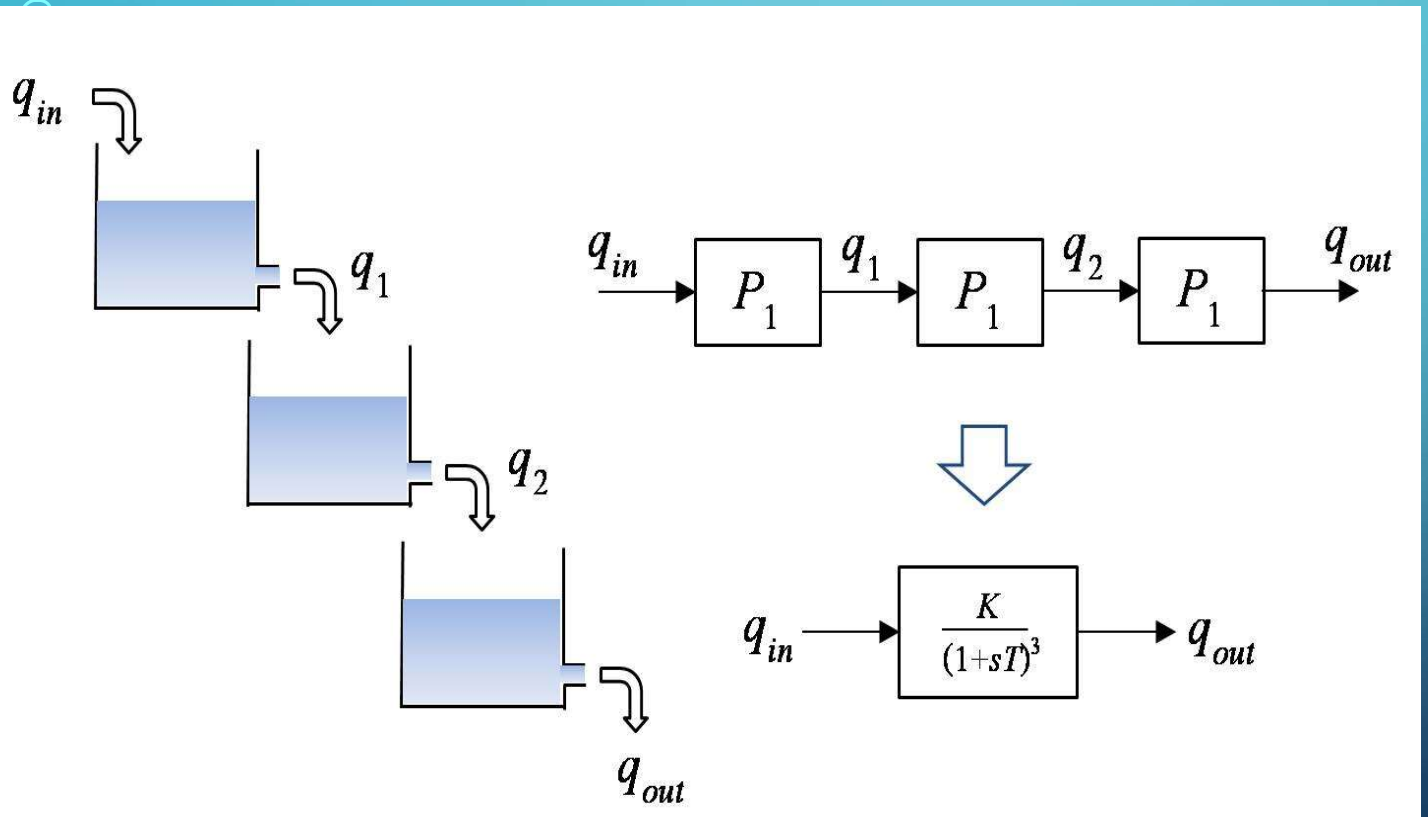
MPU6050 example

https://wokwi.com/projects/406632207671524353



The following materials may require control system background

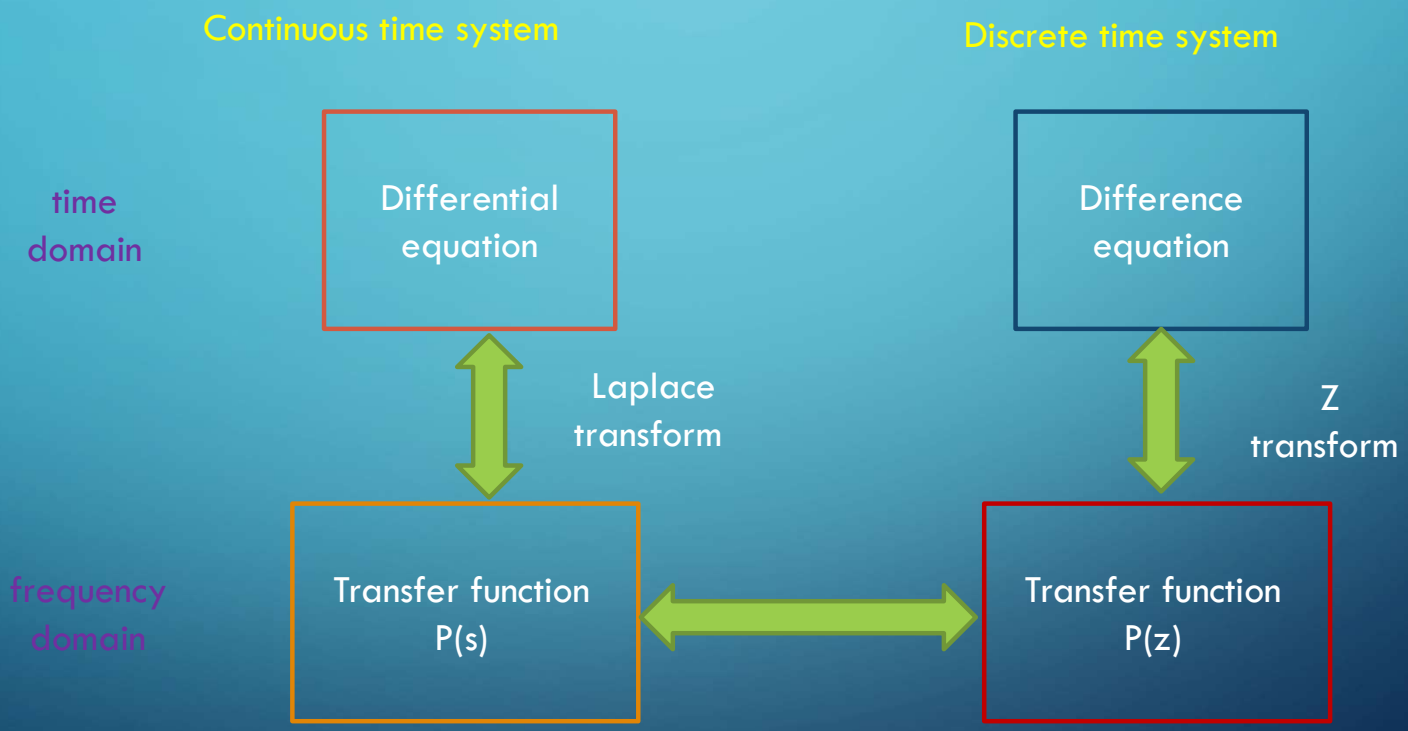
LTI SIMULATION



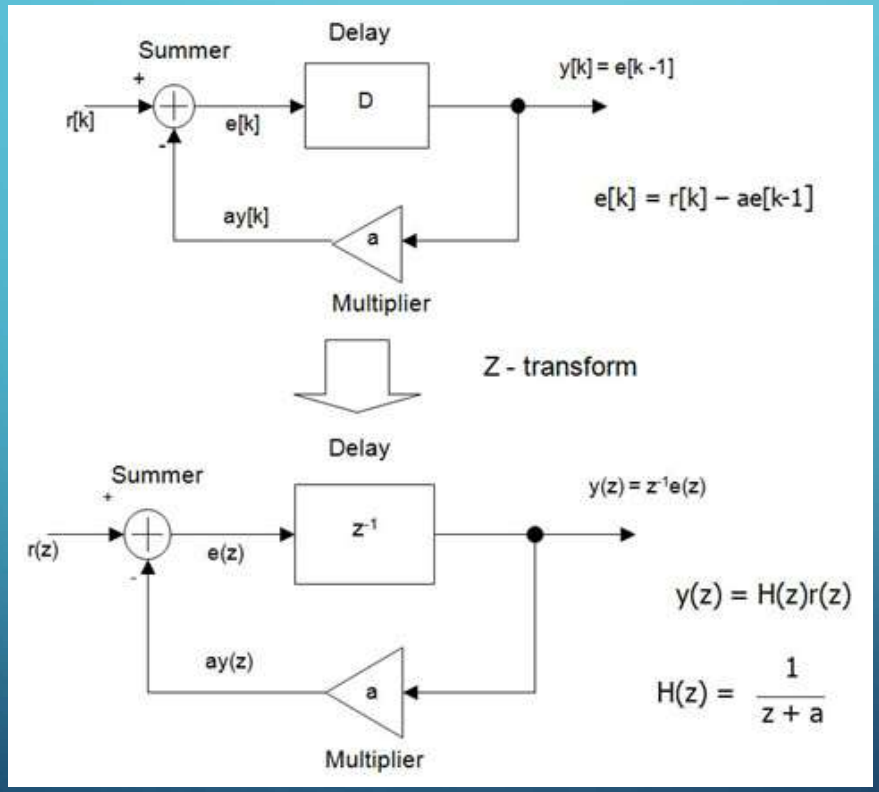
normalized transfer function

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

CONTINUOUS-DISCRETE RELATIONSHIPS



DISCRETE-TIME SYSTEM DESCRIPTION



Difference equation

Discrete-time
Transfer function

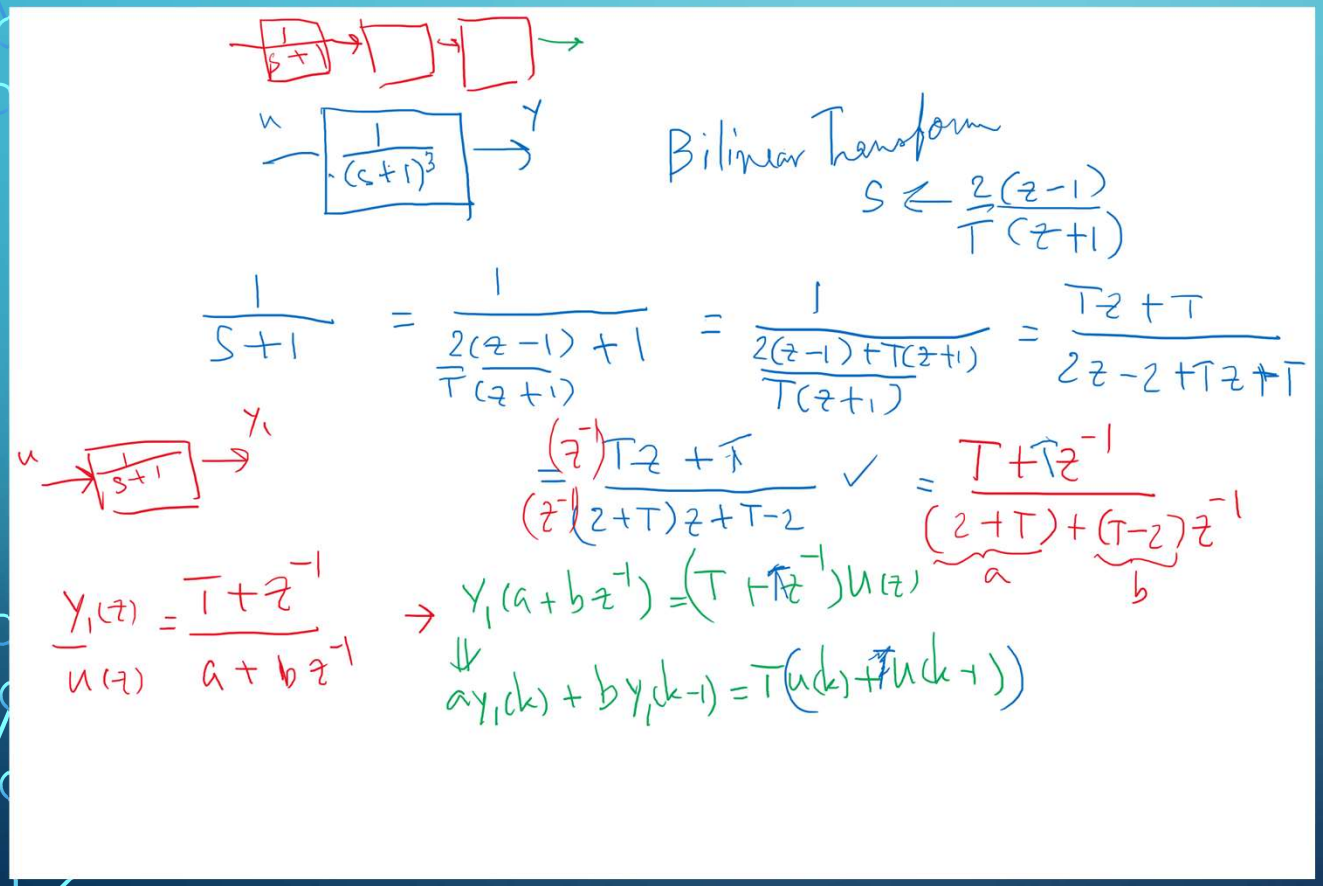
TRANSFORMATION OF C(S) TO C(Z)

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

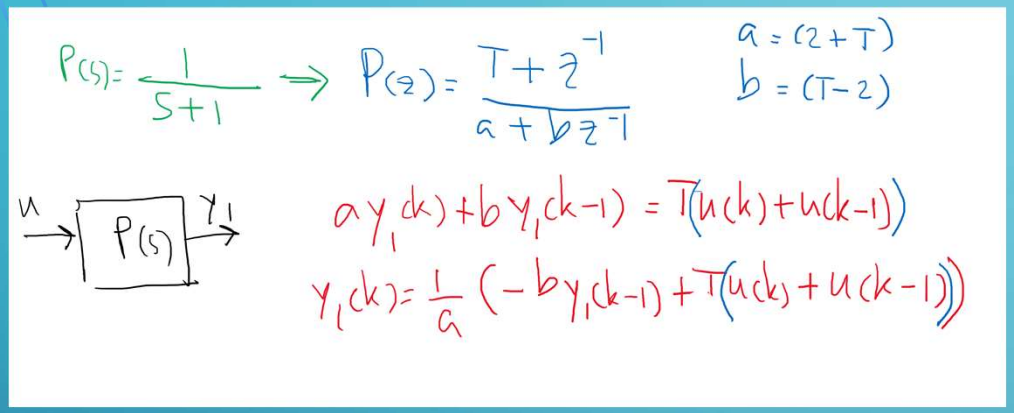
Backward difference $C(z) = C(s)\Big|_{s \to \frac{z-1}{Tz}}$

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

CONVERT 1/(S+1) TO DISCRETE TRANSFER FUNCTION AND DIFFERENCE EQUATION



SIMULATION ON JUPYTER NOTEBOOK (1)




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

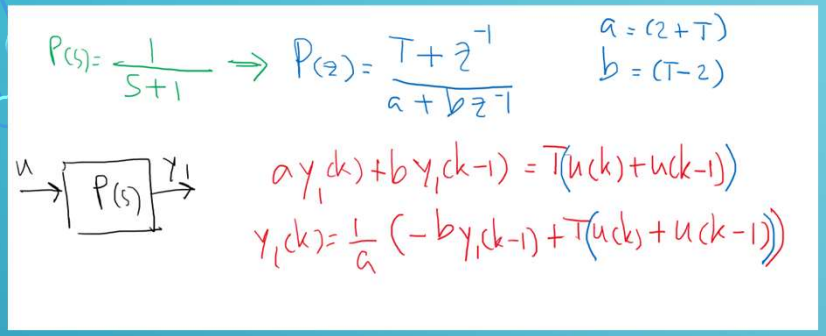
input vectors

- 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 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

SIMULATION ON JUPYTER NOTEBOOK (2)



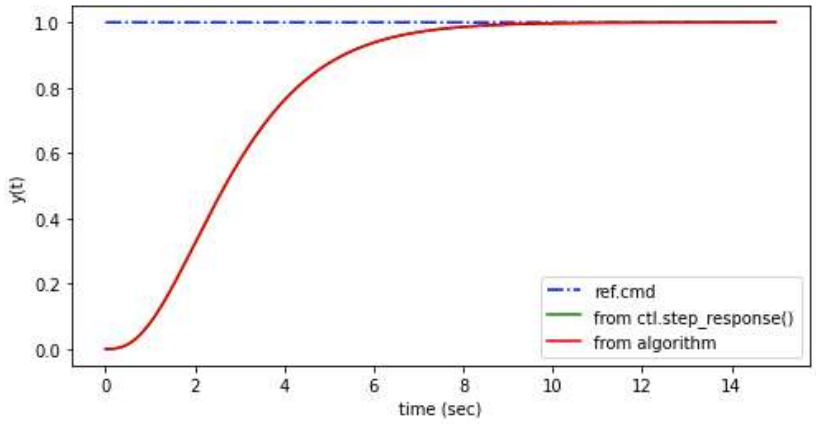

```
# lag3 simulation function
def lag3sim(uvec, tvec):
    yvec = np.zeros(tvec.shape)
    T = tvec[1] - tvec[0]
    a = 2 + T
    b = T - 2
    u_states = np.zeros((6,1))
    y_states = np.zeros((6,1))
    for i in range(len(tvec)):
        yvec[i] = lag3(a,b,T, uvec[i], u_states,
y_states)
    return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

SIMULATION ON 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()
```



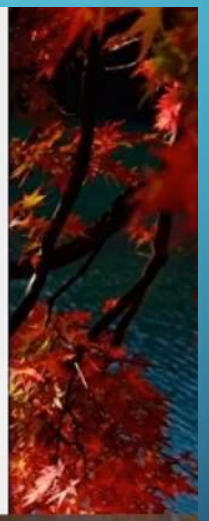
SIMULATION ON ESP32 (1)

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
```

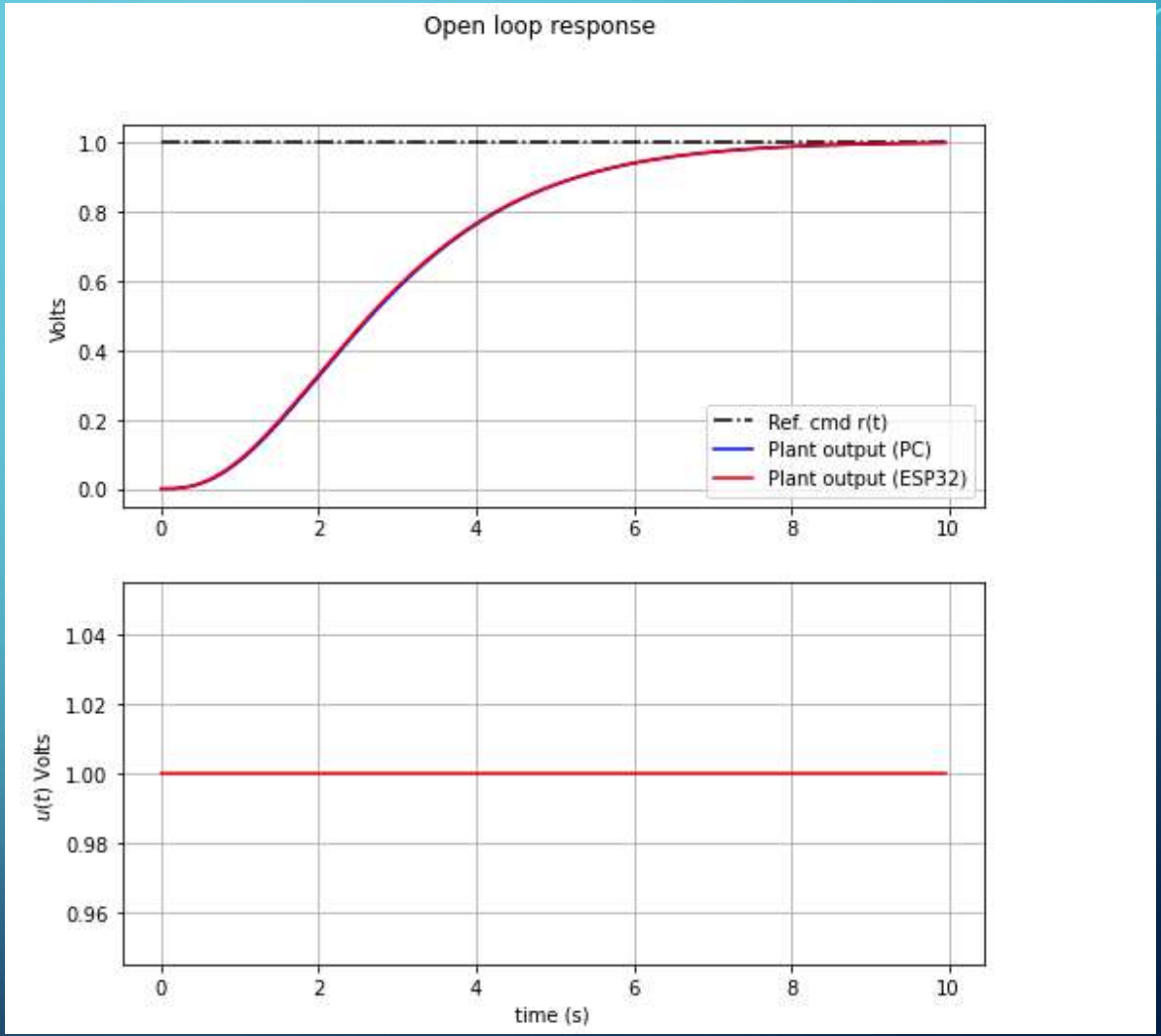
Shell

```
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],
```

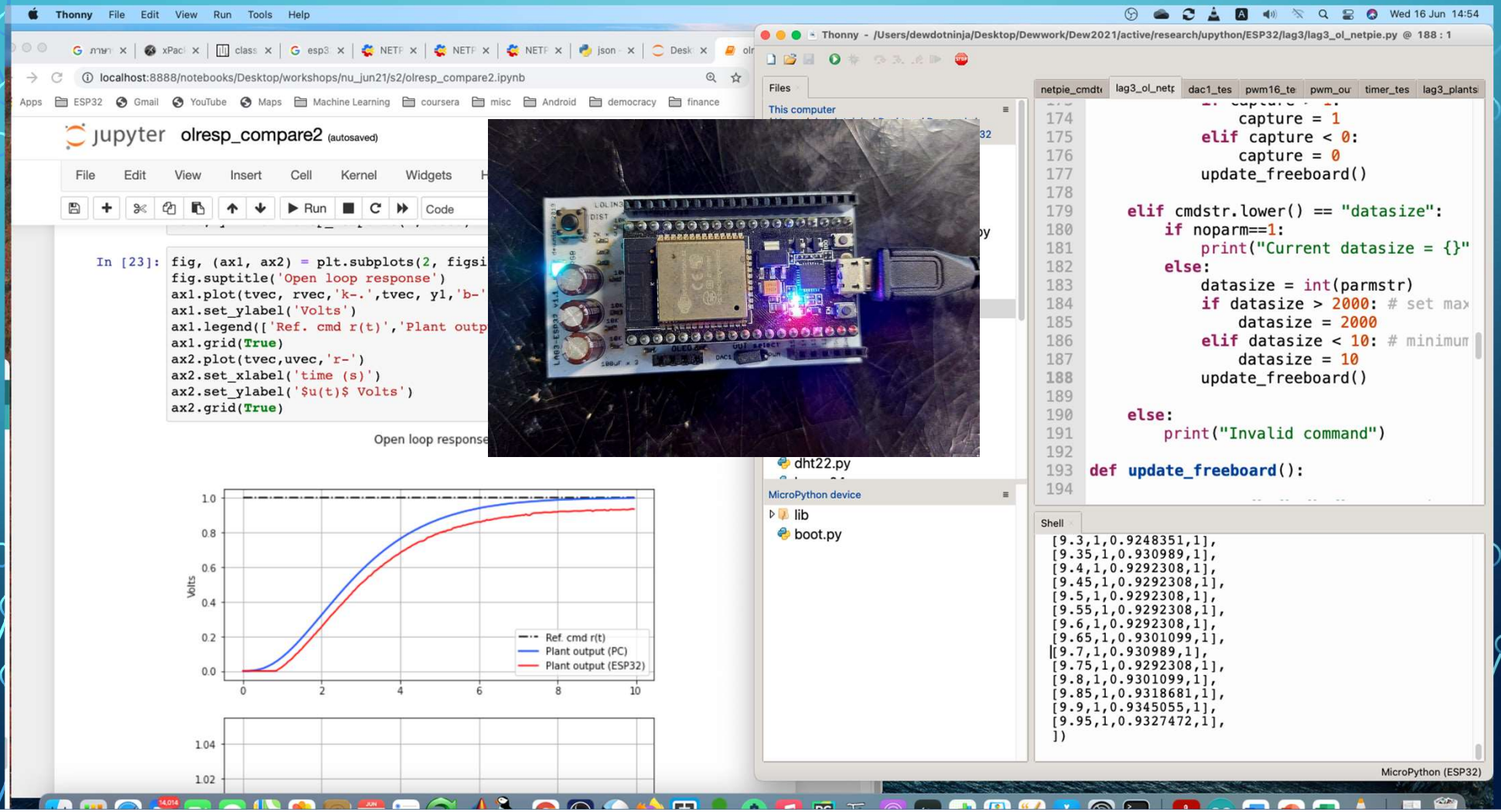


SIMULATION ON ESP32 (2)

Compare simulation between PC (Python control library) and ESP32

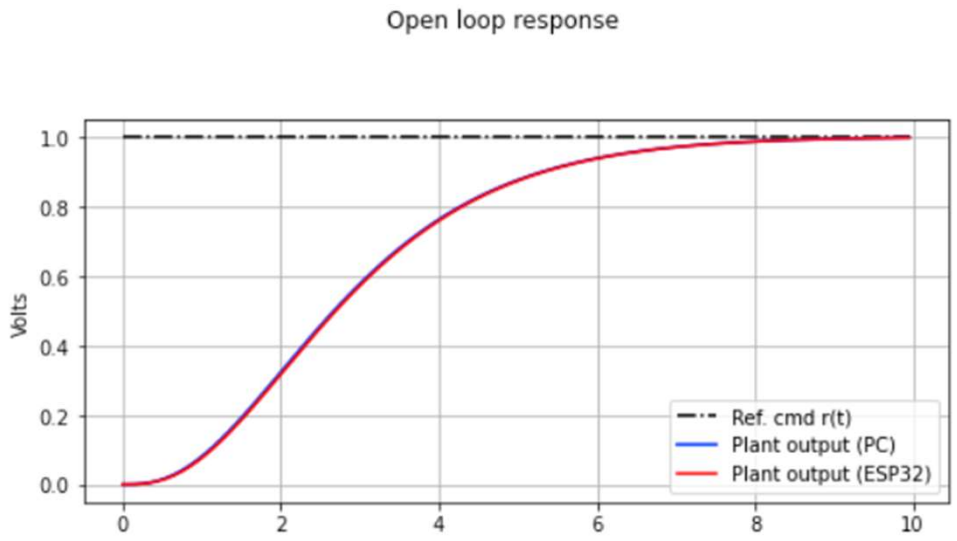


COMPARE OPEN-LOOP RESPONSES FROM LAG3



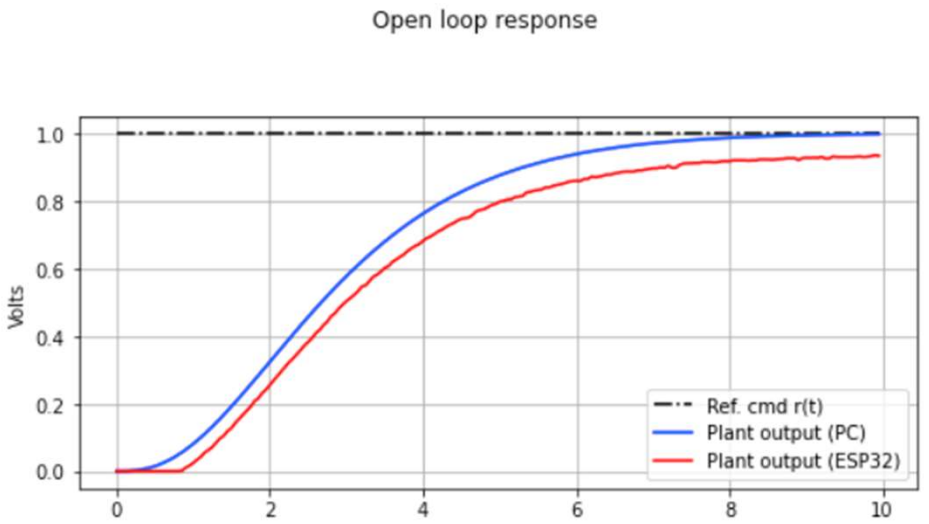
OPEN-LOOP RESPONSE SIMULATION FROM ALGORITHM

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)
```

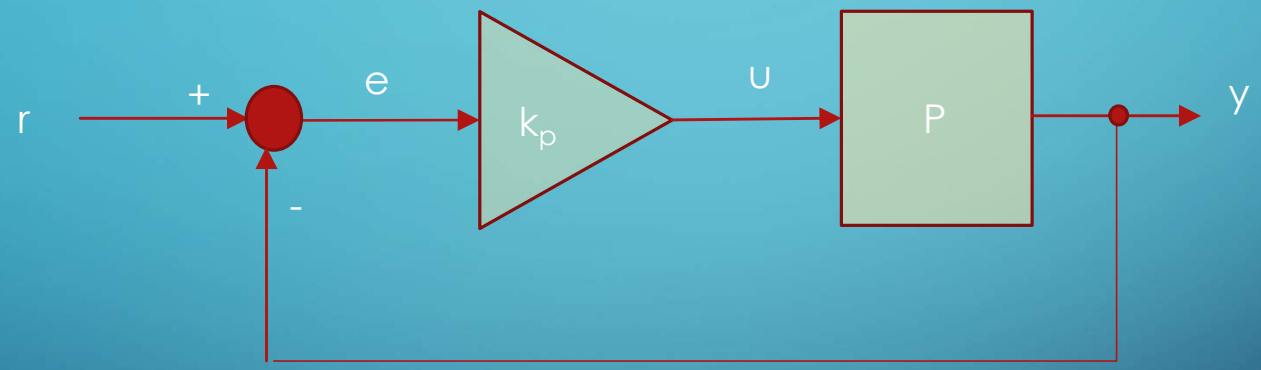


OPEN-LOOP RESPONSE FROM ELECTRONIC CIRCUITS

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)
```



PROPORTIONAL CONTROL

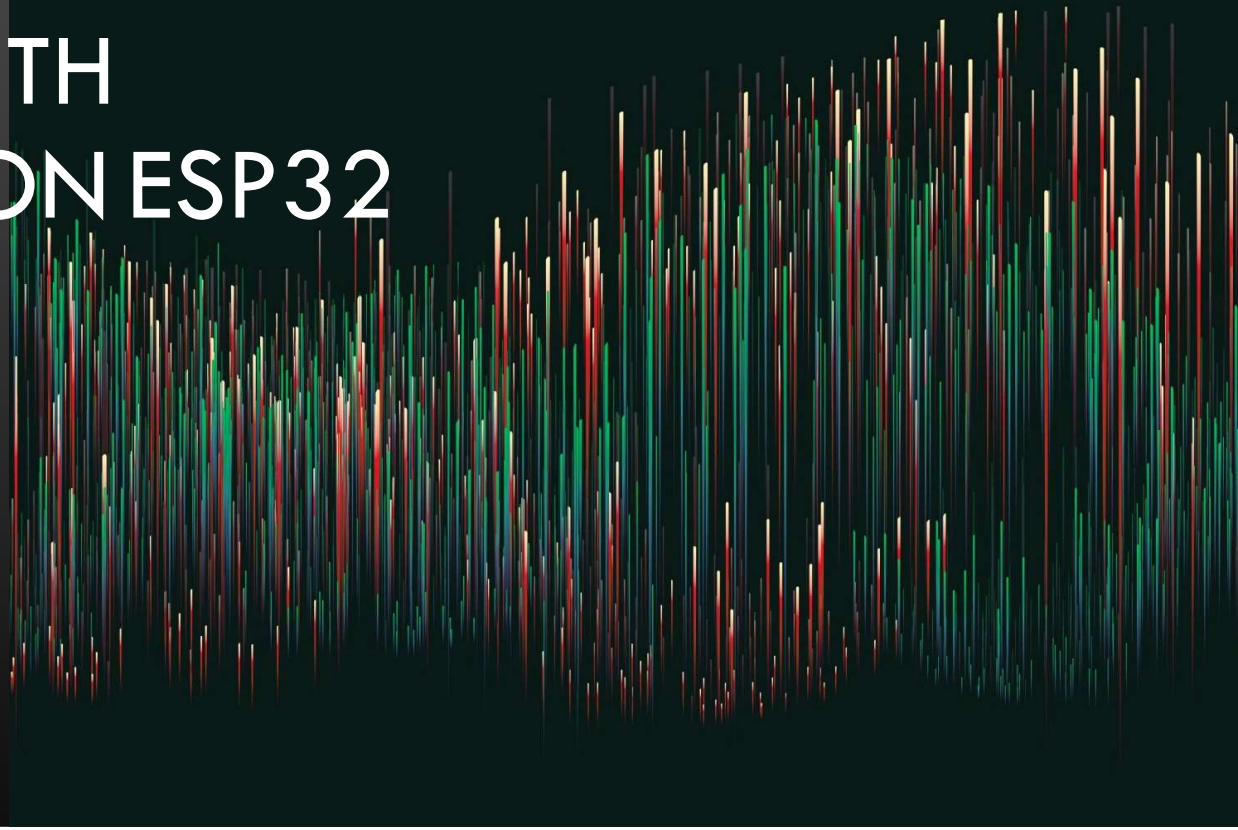


PID CONTROL WITH MICROPYTHON ON ESP32

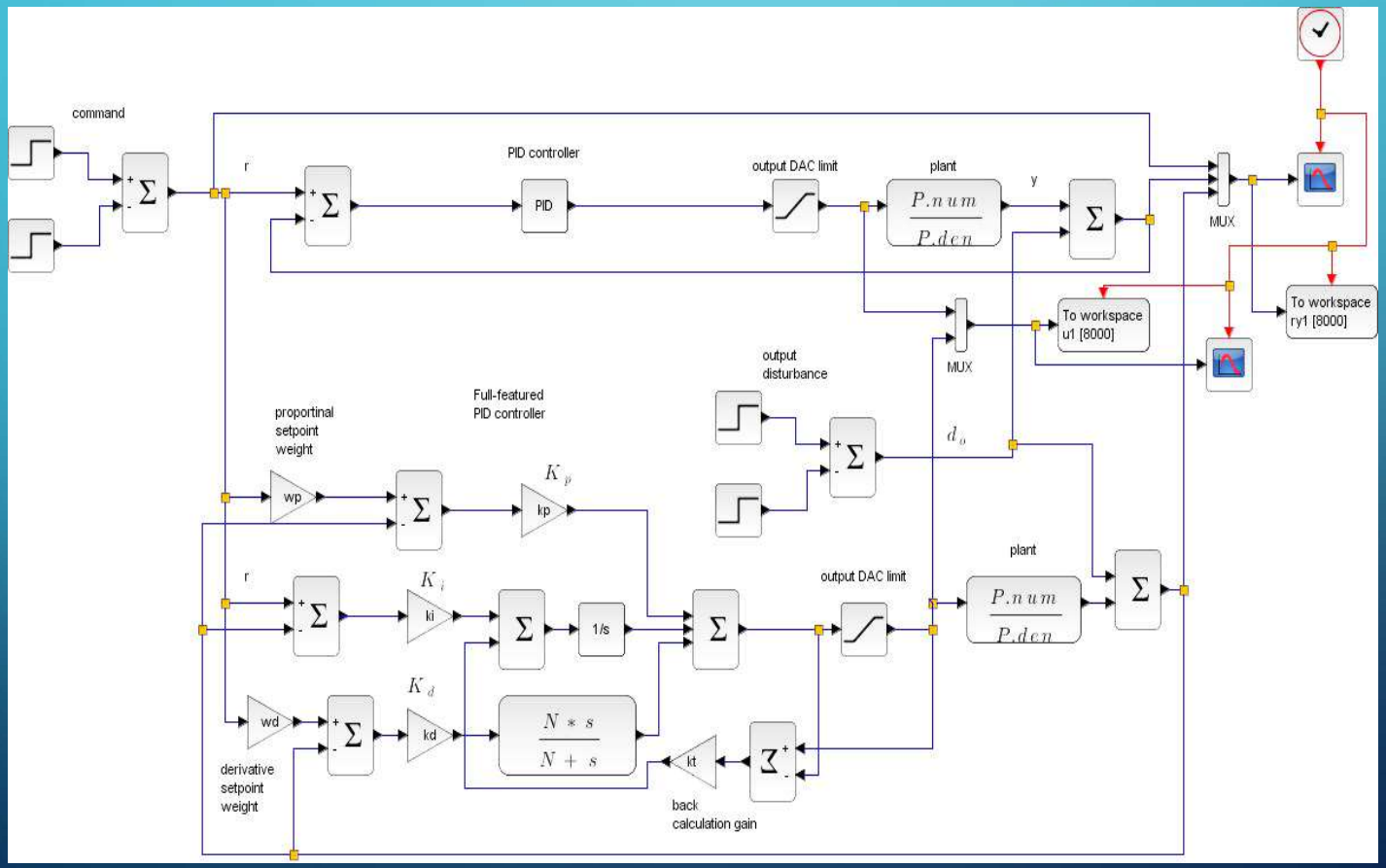
DR.VARODOM TOOCHINDA

DEPT. OF MECHANICAL ENGINEERING

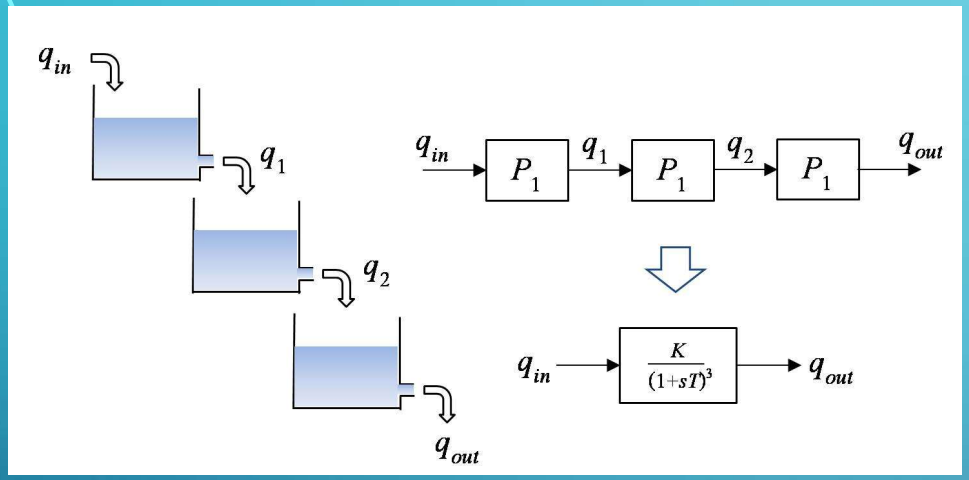
KASETSART UNIVERSITY



STRUCTURE OF FULL-FEATURED PID



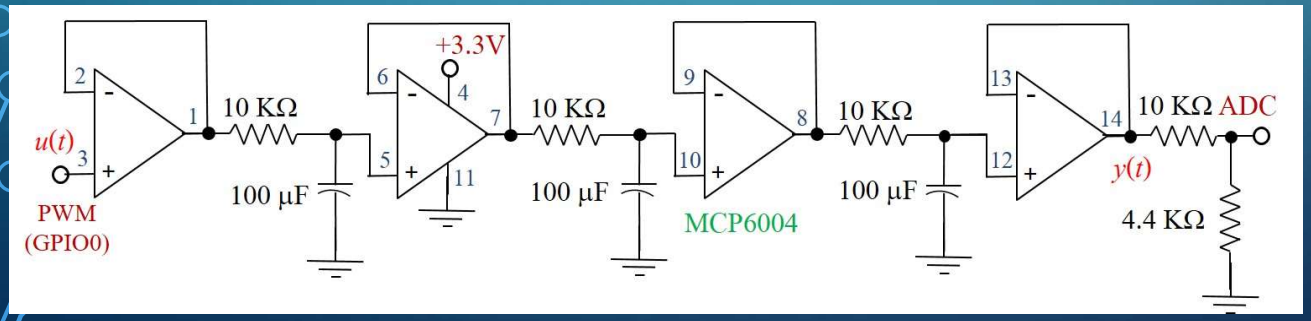
TEST ON LAG3-ESP32 (OR USING PLANT SIMULATION)

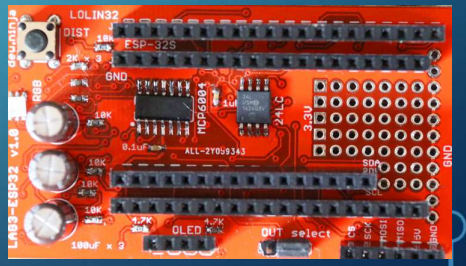


normalized transfer function

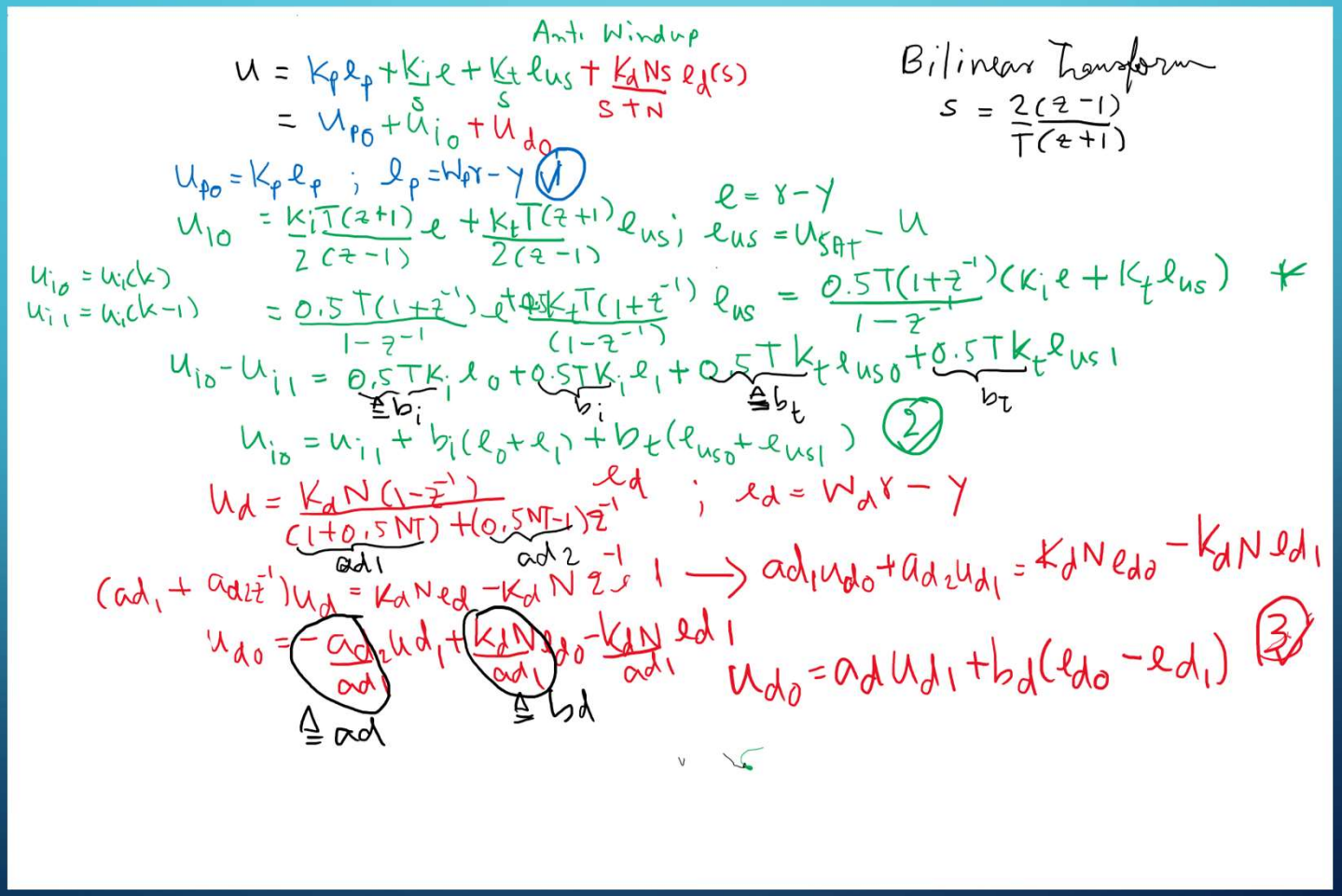
$$\frac{1}{(s+1)^3}$$

Simulated by RC and op-amp circuit





CONVERT CONTINUOUS-TIME TO DISCRETE (1)



CONVERT CONTINUOUS TIME TO DISCRETE (2)

$$u_0 = u_{p0} + u_{i0} + u_{d0}$$

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

$$u_{i0} = u_{i1} + b_i (e_0 + e_1) + b_t (e_{us0} + e_{us1}) \quad ; \; e = r - y \quad ; \; e_{us} = u_{sat} - u$$

$$u_{d0} = a_{d1} u_{d1} + b_d (e_{d0} - e_{d1}) \quad ; \; e_d = w_d r - y$$

$$b_i = 0.5 T K_i$$

$$b_t = 0.5 T K_t$$

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

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

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

FUNCTION TO COMPUTE COEFFICIENTS

```
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()
```

PID CONTROLLER FUNCTION

```
def PID_controller():
    global e1,e0,ed1,ed0, eus1,eus0,
   ui1, 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
```

Call function in infinite loop (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
```

add commands in 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":
        :
```

copy and paste step response to Jupyter notebook

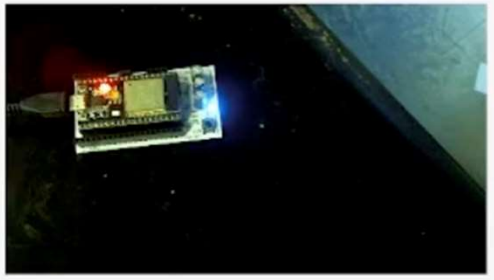
jupyter plot_pid_stepcompare Last Checkpoint: 16 minutes ago (unsaved changes)

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

Run Markdown

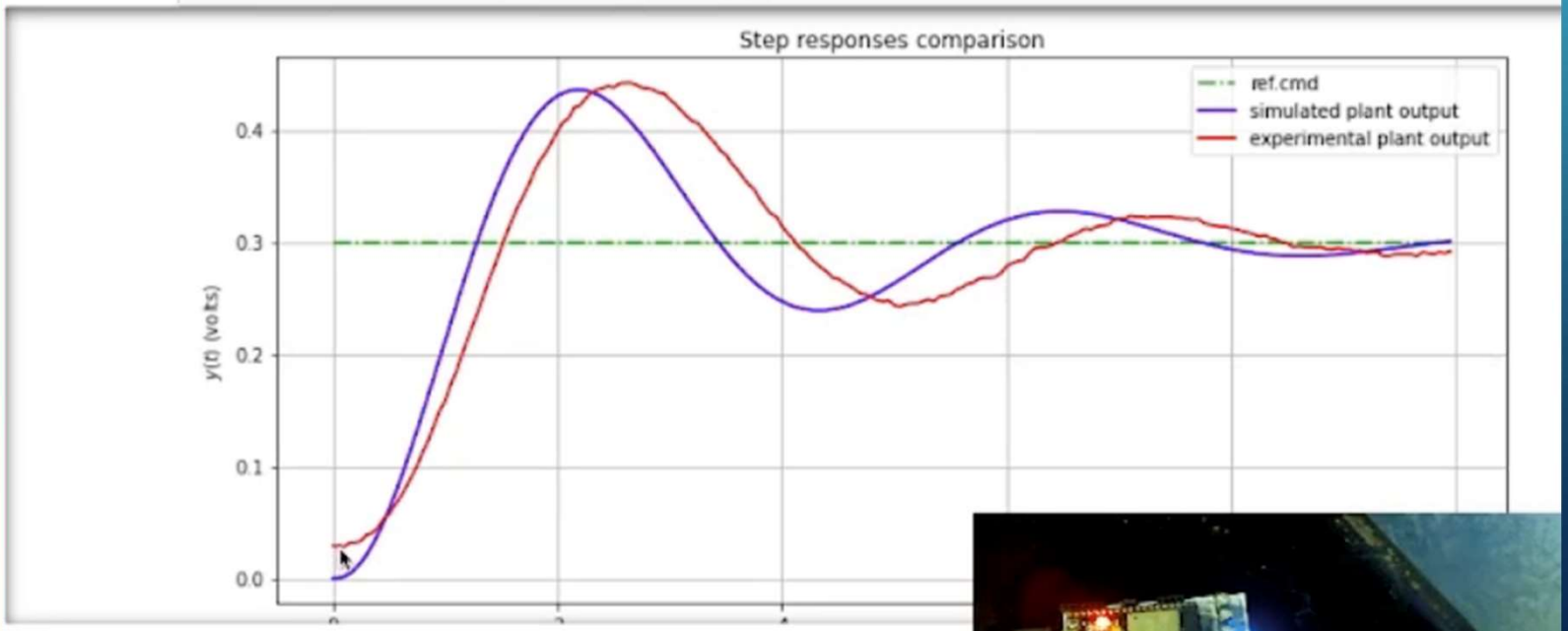
In [34]:
```
import numpy as np
import matplotlib.pyplot as plt
import control as ctl
```

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],
```



step response comparison

```
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)
```

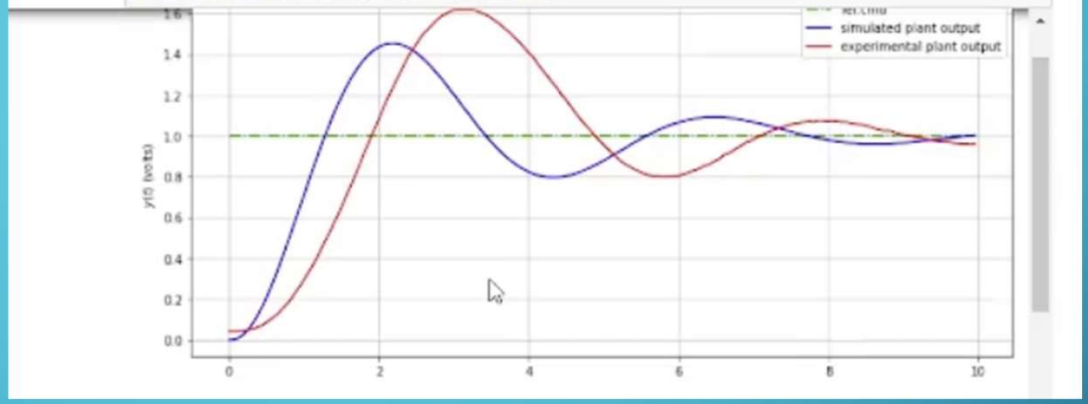


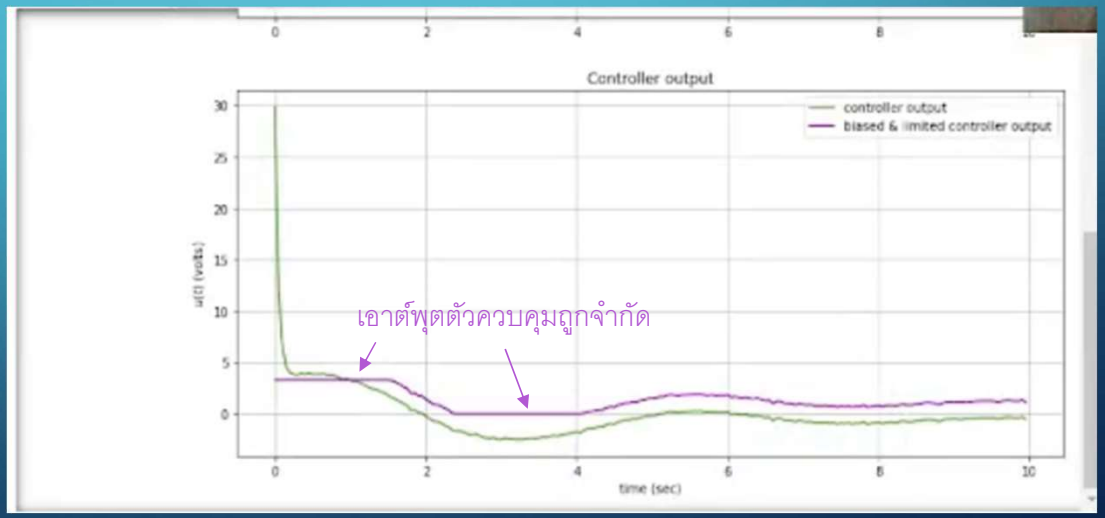
```
In [ ]:
```

```
In [ ]:
```

saturation effect when setting ref.cmd = 1.0

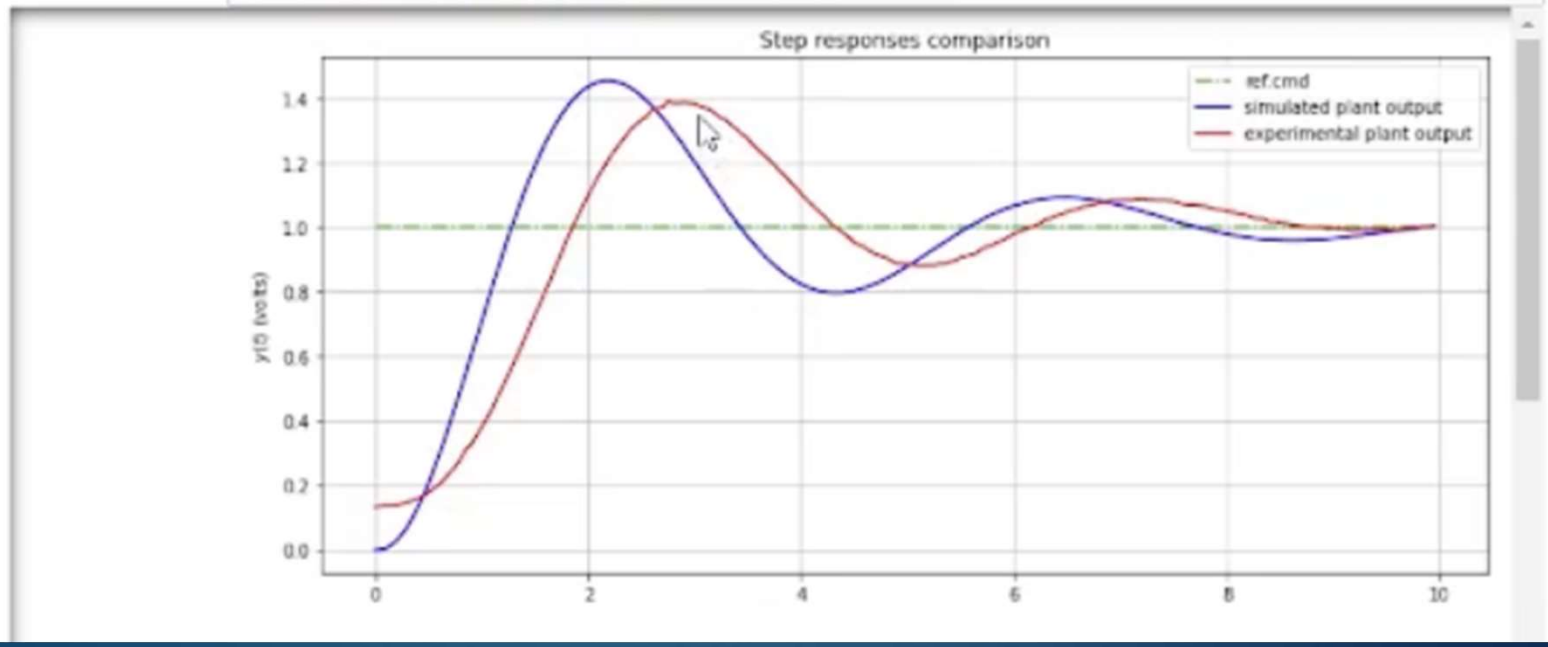
```
In [6]: # use the same parameters as in the ESP32
        kp = 4.8
        ki = 2.74
        kd = 2.1
        pid_compare(kp,ki,kd,datamat,scale=1)
```



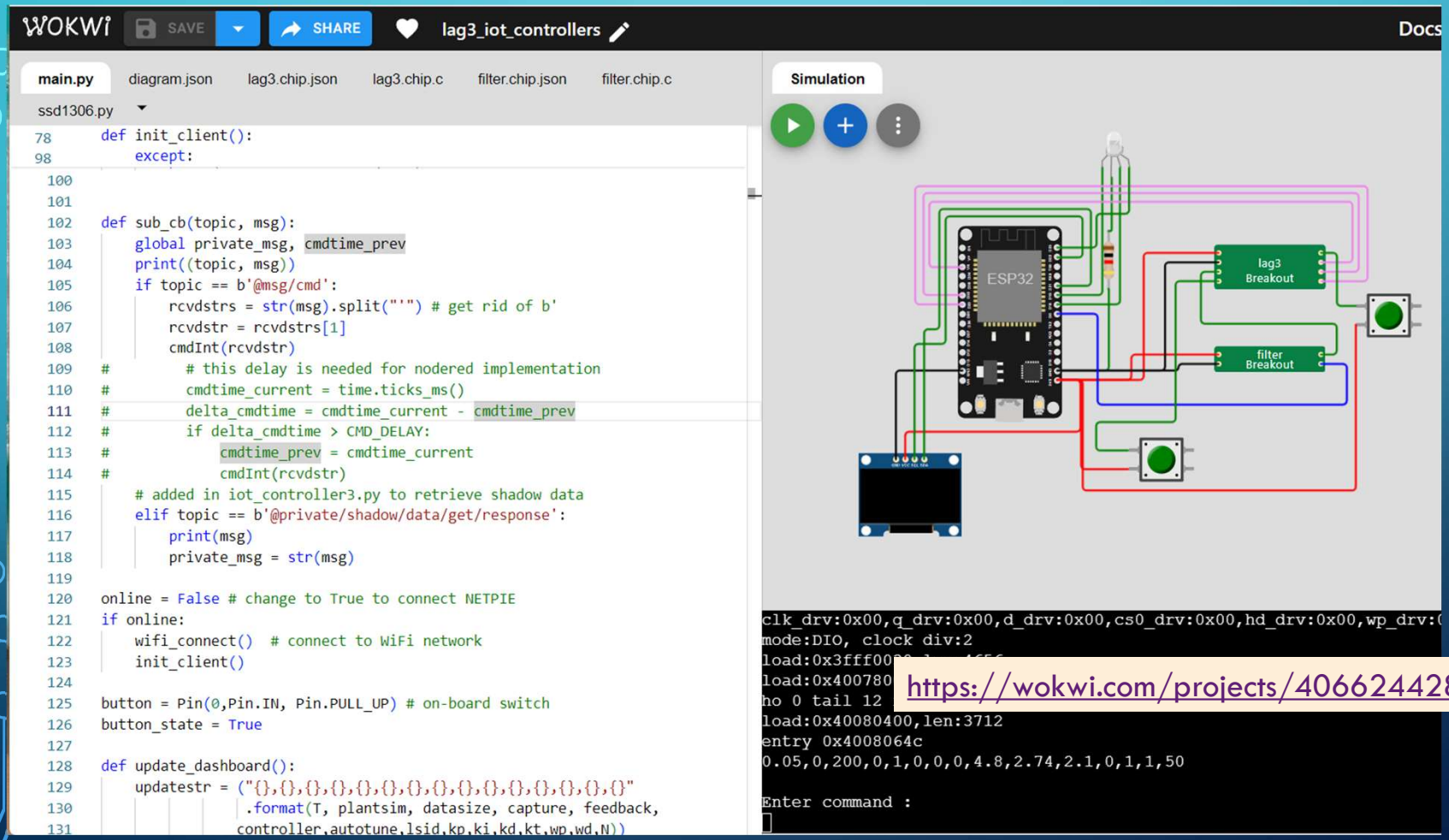


compensate with back calculation anti-windup (kt=0.5)

```
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)
```



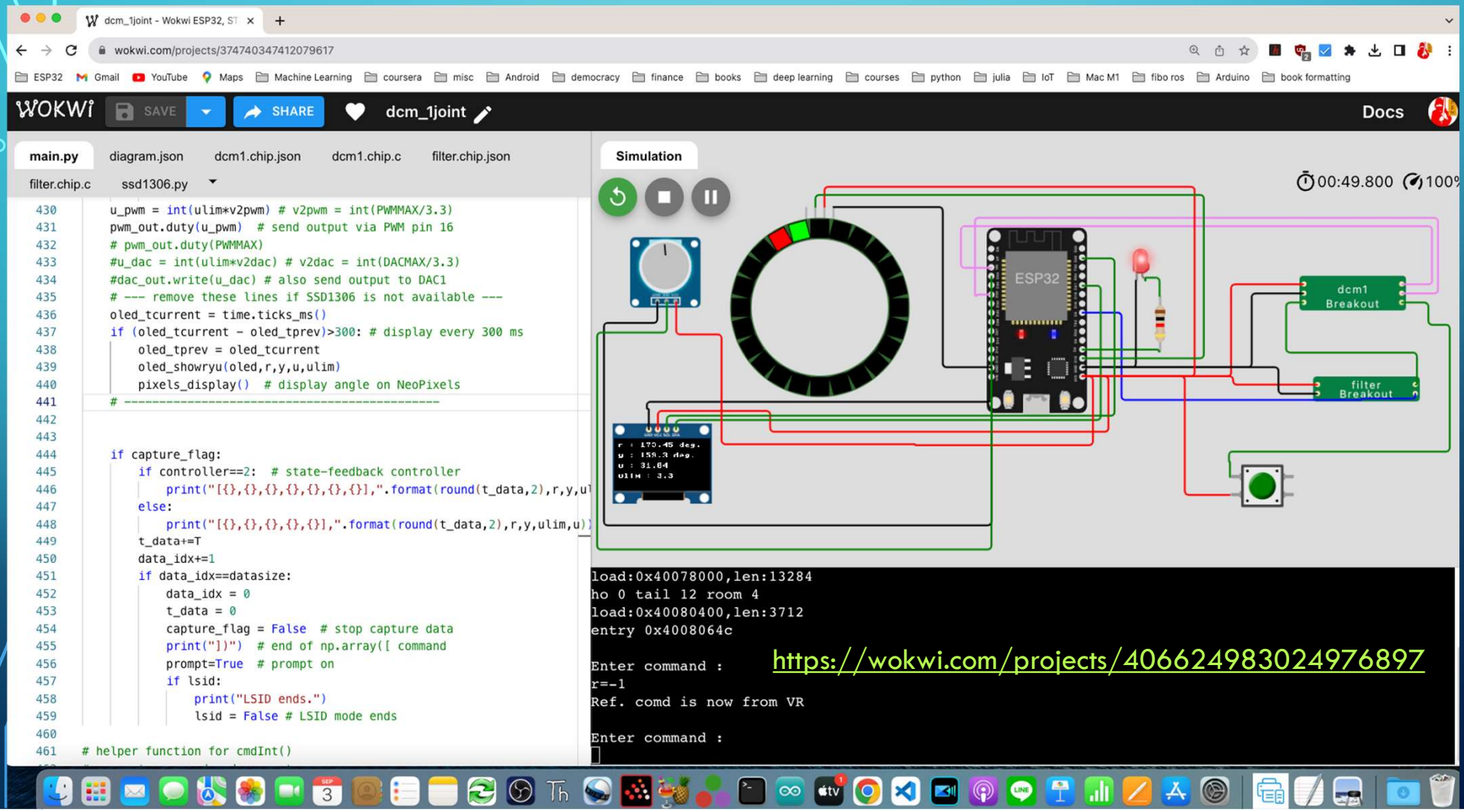
LAG3 SIMULATION ON WOKWI




```
78      def init_client():
98          except:
100
101
102     def sub_cb(topic, msg):
103         global private_msg, cmdtime_prev
104         print((topic, msg))
105         if topic == b'@msg/cmd':
106             rcvdstrs = str(msg).split("'") # get rid of b'
107             rcvdstr = rcvdstrs[1]
108             cmdInt(rcvdstr)
109     #         # this delay is needed for nodered implementation
110     #         cmdtime_current = time.ticks_ms()
111     #         delta_cmdtime = cmdtime_current - cmdtime_prev
112     #         if delta_cmdtime > CMD_DELAY:
113     #             cmdtime_prev = cmdtime_current
114     #             cmdInt(rcvdstr)
115         # added in iot_controller3.py to retrieve shadow data
116         elif topic == b'@private/shadow/data/get/response':
117             print(msg)
118             private_msg = str(msg)
119
120     online = False # change to True to connect NETPIE
121     if online:
122         wifi_connect()  # connect to WiFi network
123         init_client()
124
125     button = Pin(0,Pin.IN, Pin.PULL_UP) # on-board switch
126     button_state = True
127
128     def update_dashboard():
129         updatestr = ("{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}"
130                      .format(T, plantsim, datasize, capture, feedback,
131                      controller,autotune,lsid,kp,ki,kd,kt,wp,wd,N))
```

https://wokwi.com/projects/406624428008336385

SIMULATION EXAMPLE



Thank You