Programmes en Micropython SmartHome

Test Led pin 12

exemple001.py

```
from machine import Pin
import time

led = Pin(12, Pin.OUT)# Build an LED object, connect the external LED
light to pin 0, and set pin 0 to output mode
while True:
    led.value(1)# turn on led
    time.sleep(1)# delay 1s
    led.value(0)# turn off led
    time.sleep(1)# delay 1s
```

Test Led PWM

exemple002.py

```
import time
from machine import Pin,PWM

#The way that the ESP32 PWM pins output is different from traditionally
controllers.
#It can change frequency and duty cycle by configuring PWM's parameters
at the initialization stage.
#Define GPIO 0's output frequency as 10000Hz and its duty cycle as 0,
and assign them to PWM.
p0 = Pin(012, Pin.OUT)
pwm = PWM(p0, freq=10000, duty_u16=8192)


try:
    while True:
#The range of duty cycle is 0-1023, so we use the first for loop to
control PWM to change the duty
#cycle value,making PWM output 0% -100%; Use the second for loop to
make PWM output 100%-0%.
        for i in range(0,1023):
            pwm.duty(i)
            time.sleep_ms(1)

        for i in range(0,1023):
```

```
            pwm.duty(1023-i)
            time.sleep_ms(1)
except:
#Each time PWM is used, the hardware Timer will be turned ON to
cooperate it. Therefore, after each use of PWM,
#deinit() needs to be called to turned OFF the timer. Otherwise, the
PWM may fail to work next time.
    pwm.deinit()
```

Test des 2 Boutons

exemple003.py

```
from machine import Pin
import time

button1 = Pin(16, Pin.IN, Pin.PULL_UP)
button2 = Pin(27, Pin.IN, Pin.PULL_UP)

while True:
    btnVal1 = button1.value()  # Reads the value of button 1
    btnVal2 = button2.value()
    print("button1 =",btnVal1)  #Print it out in the shell
    print("button2 =",btnVal2)
    time.sleep(0.1) #delay 0.1s
```

Test Bouton 1 M/A Led

exempl004.py

```
from machine import Pin
import time

button1 = Pin(16, Pin.IN, Pin.PULL_UP)
led = Pin(12, Pin.OUT)
count = 0

while True:
    btnVal1 = button1.value()  # Reads the value of button 1
    #print("button1 =",btnVal1)  #Print it out in the shell
    if(btnVal1 == 0):
        time.sleep(0.01)
        while(btnVal1 == 0):
            btnVal1 = button1.value()
```

```
            if(btnVal1 == 1):
                count = count + 1
                print(count)
    val = count % 2
    if(val == 1):
        led.value(1)
    else:
        led.value(0)
    time.sleep(0.1) #delay 0.1s
```

Test PIR : detection de personnes

exemple005.py

```
from machine import Pin
import time

PIR = Pin(14, Pin.IN)
while True:
    value = PIR.value()
    print(value, end = " ")
    if value == 1:
        print("Des personnes sont dans la zone ...!")
    else:
        print("il n'y a personne ...!")
    time.sleep(0.1)
```

Test Buzzer Music

exemple006.py

```
from machine import Pin, PWM
from time import sleep
buzzer = PWM(Pin(25))

buzzer.duty(1000)

# Happy birthday
buzzer.freq(294)
sleep(0.25)
buzzer.freq(440)
sleep(0.25)
buzzer.freq(392)
sleep(0.25)
buzzer.freq(532)
```

```
sleep(0.25)
buzzer.freq(494)
sleep(0.25)
buzzer.freq(392)
sleep(0.25)
buzzer.freq(440)
sleep(0.25)
buzzer.freq(392)
sleep(0.25)
buzzer.freq(587)
sleep(0.25)
buzzer.freq(532)
sleep(0.25)
buzzer.freq(392)
sleep(0.25)
buzzer.freq(784)
sleep(0.25)
buzzer.freq(659)
sleep(0.25)
buzzer.freq(532)
sleep(0.25)
buzzer.freq(494)
sleep(0.25)
buzzer.freq(440)
sleep(0.25)
buzzer.freq(698)
sleep(0.25)
buzzer.freq(659)
sleep(0.25)
buzzer.freq(532)
sleep(0.25)
buzzer.freq(587)
sleep(0.25)
buzzer.freq(532)
sleep(0.5)
buzzer.duty(0)
```

Test Servo Porte entrée : angle 25° 77° 180°

exemple007.py

```
from machine import Pin, PWM
import time
pwm = PWM(Pin(13))
pwm.freq(50)

'''
```

```
Duty cycle corresponding to the Angle
0°----2.5%----25
45°----5%----51.2
90°----7.5%----77
135°----10%----102.4
180°----12.5%----128
'''
angle_0 = 25
angle_90 = 77
angle_180 = 128

while True:
    pwm.duty(angle_0)
    time.sleep(1)
    pwm.duty(angle_90)
    time.sleep(1)
    pwm.duty(angle_180)
    time.sleep(1)
```

Test Capteur Humidité 0= Fenetre Ouverte, 0> Fenetre fermée

exemple008.py

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC,PWM
import time
pwm = PWM(Pin(5))
pwm.freq(50)

# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(34))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)

# Read ADC value once every 0.1seconds, convert ADC value to DAC value
and output it,
# and print these data to “Shell”.
try:
    while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC
Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        if(voltage > 0.6):
            pwm.duty(46)
        else:
            pwm.duty(100)
```

```
        time.sleep(0.1)
except:
    pass
```

Test Neopixels : Blanc, Rouge, Vert, Bleu

exemple009.py

```
#Import Pin, neopiexl and time modules.
from machine import Pin
import neopixel
import time

#Define the number of pin and LEDs connected to neopixel.
pin = Pin(26, Pin.OUT)
np = neopixel.NeoPixel(pin, 4)

#brightness :0-255
brightness=100
colors=[[brightness,0,0],                    #red
        [0,brightness,0],                    #green
        [0,0,brightness],                    #blue
        [brightness,brightness,brightness],  #white
        [0,0,0]]                             #close

#Nest two for loops to make the module repeatedly display five states
of red, green, blue, white and OFF.
while True:
    for i in range(0,5):
        for j in range(0,4):
            np[j]=colors[i]
            np.write()
            time.sleep_ms(50)
        time.sleep_ms(500)
    time.sleep_ms(500)
```

Test Bouton et Neopixels Bp2 + incremente les couleurs Bp1 - decrementre les couleurs

exemple010.py

```
#Import Pin, neopiexl and time modules.
from machine import Pin
import neopixel
```

```
import time

button1 = Pin(16, Pin.IN, Pin.PULL_UP)
button2 = Pin(27, Pin.IN, Pin.PULL_UP)
count = 0

#Define the number of pin and LEDs connected to neopixel.
pin = Pin(26, Pin.OUT)
np = neopixel.NeoPixel(pin, 4)

#brightness :0-255
brightness=100
colors=[[0,0,0],
        [brightness,0,0],                    #red
        [0,brightness,0],                    #green
        [0,0,brightness],                    #blue
        [brightness,brightness,brightness]  #white
        ]                                 #close

def func_color(val):
    for j in range(0,4):
        np[j]=colors[val]
        np.write()
        time.sleep_ms(50)

#Nest two for loops to make the module repeatedly display five states
of red, green, blue, white and OFF.
while True:
    btnVal1 = button1.value()  # Reads the value of button 1
    #print("button1 =",btnVal1)  #Print it out in the shell
    if(btnVal1 == 0):
        time.sleep(0.01)
        while(btnVal1 == 0):
            btnVal1 = button1.value()
            if(btnVal1 == 1):
                count = count - 1
                print(count)
                if(count <= 0):
                    count = 0

    btnVal2 = button2.value()
    if(btnVal2 == 0):
        time.sleep(0.01)
        while(btnVal2 == 0):
            btnVal2 = button2.value()
            if(btnVal2 == 1):
                count = count + 1
                print(count)
                if(count >= 4):
                    count = 4
```

```
    if(count == 0):
        func_color(0)
    elif(count == 1):
        func_color(1)
    elif(count == 2):
        func_color(2)
    elif(count == 3):
        func_color(3)
    elif(count == 4):
        func_color(4)
```

Test ventilateur dans les 2 sens

exemple011.py

```
from machine import Pin,PWM
import time
#Two pins of the motor
p0 = Pin(19,Pin.OUT)
p1 = Pin(18,Pin.OUT)
#INA =PWM(po,10000,0)#INA corresponds to IN+
#INB =PWM(p1,10000,2)#INB corresponds to IN-
INA = PWM(p0, freq=10000, duty_u16=8192)
INB = PWM(p1, freq=10000, duty_u16=8192)


try:
    while True:
        #Counterclockwise 4s
        INA.duty(0) #The range of duty cycle is 0-1023
        INB.duty(700)
        time.sleep(4)
        #stop 2s
        INA.duty(0)
        INB.duty(0)
        time.sleep(2)
        #Turn clockwise for 4s
        INA.duty(600)
        INB.duty(0)
        time.sleep(4)
        #stop 2s
        INA.duty(0)
        INB.duty(0)
        time.sleep(2)
except:
    INA.duty(0)
    INB.duty(0)
```

```
    INA.deinit()
    INB.deinit()
```

Bouton1 commande ventilateur

exemple012.py

```
from machine import Pin,PWM
import time
#Two pins of the motor
p0 = Pin(19,Pin.OUT)
p1 = Pin(18,Pin.OUT)
INA =PWM(p0,freq=10000, duty_u16=8192)#INA corresponds to IN+
INB =PWM(p1,freq=10000, duty_u16=8192)#INB corresponds to IN-

button1 = Pin(16, Pin.IN, Pin.PULL_UP)
count1 = 0


try:
    while True:
        btnVal1 = button1.value()  # Reads the value of button 1
        if(btnVal1 == 0):
            time.sleep(0.01)
            while(btnVal1 == 0):
                btnVal1 = button1.value()
                if(btnVal1 == 1):
                    count1 = count1 + 1
                    print("comp1" , count1)
        val1 = count1 % 2
        if(val1 == 1):
            INA.duty(0) #The range of duty cycle is 0-1023
            INB.duty(700)
        else:
            INA.duty(0)
            INB.duty(0)
except:
    INA.duty(0)
    INB.duty(0)
    INA.deinit()
    INB.deinit()
```

Test capteur Gaz et affichage sur LCD

exemple013.py

```
from time import sleep_ms, ticks_ms
from machine import I2C, Pin
from i2c_lcd import I2cLcd

DEFAULT_I2C_ADDR = 0x27

i2c = I2C(scl=Pin(22), sda=Pin(21), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)

from machine import Pin
import time
gas = Pin(23, Pin.IN, Pin.PULL_UP)

while True:
    gasVal = gas.value()  # Reads the value of button 1
    print("gas =",gasVal)  #Print it out in the shell
    lcd.move_to(1, 1)
    lcd.putstr('val: {}'.format(gasVal))
    if(gasVal == 1):
        #lcd.clear()
        lcd.move_to(1, 0)
        lcd.putstr('Safety       ')
    else:
        lcd.move_to(1, 0)
        lcd.putstr('dangerous')
    time.sleep(0.1) #delay 0.1s
```

Test DHT11 et affichage sur LCD

exemple014.py

```
# Import machine, time and dht modules.
import machine
import time
import dht
from time import sleep_ms, ticks_ms
from machine import I2C, Pin
from i2c_lcd import I2cLcd

#Associate DHT11 with Pin(17).
DHT = dht.DHT11(machine.Pin(17))

DEFAULT_I2C_ADDR = 0x27

i2c = I2C(scl=Pin(22), sda=Pin(21), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
```

```
while True:
    DHT.measure() # Start DHT11 to measure data once.
   # Call the built-in function of DHT to obtain temperature
   # and humidity data and print them in “Shell”.
print('temperature:',DHT.temperature(),'□','humidity:',DHT.humidity(),'
%')
    lcd.move_to(1, 0)
    lcd.putstr('T= {}'.format(DHT.temperature()))
    lcd.move_to(1, 1)
    lcd.putstr('H= {}'.format(DHT.humidity()))
    time.sleep_ms(1000)
```

Test ouverture porte servoMoteur avec code -.- sur BP1

exemple015.py

```
# Import machine, time and dht modules.
from machine import Pin, PWM
from time import sleep_ms, ticks_ms
from machine import I2C, Pin
from i2c_lcd import I2cLcd

DEFAULT_I2C_ADDR = 0x27

i2c = I2C(scl=Pin(22), sda=Pin(21), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)

button1 = Pin(16, Pin.IN, Pin.PULL_UP)
button2 = Pin(27, Pin.IN, Pin.PULL_UP)
count = 0
time_count = 0
password = ""   #Enter password
correct_password = "-.-"  #Correct password
lcd.putstr("Enter password")
pwm = PWM(Pin(13))
pwm.freq(50)

while True:
    btnVal1 = button1.value()  # Read the value of button 1
    if(btnVal1 == 0):
        sleep_ms(10)
        while(btnVal1 == 0):
            time_count = time_count + 1  #Start counting the pressed
time of the button
            sleep_ms(200)                #The time is 200ms cumulative
            btnVal1 = button1.value()
            if(btnVal1 == 1):
```

```
                count = count + 1
                print(count)
                print(time_count)
                if(time_count > 3):      #If the pressed time of the
button is more than 200*3ms□add"-" to  password
                    lcd.clear()
                    #lcd.move_to(1, 1)
                    password = password + "-"
                else:
                    lcd.clear()
                    password = password + "."  #Otherwise add "."
                lcd.putstr('{}'.format(password))
                time_count = 0

    btnVal2 = button2.value()
    if(btnVal2 == 0):
        if(password == correct_password):  #If the password is correct
            lcd.clear()
            lcd.putstr("open")
            pwm.duty(128)  #Open the door
            password = ""  #Remove the password
            sleep_ms(1000)
        else:              #If the password is wrong
            lcd.clear()
            lcd.putstr("error")
            pwm.duty(25)  #Close the door
            sleep_ms(2000)
            lcd.clear()
            lcd.putstr("enter again")
            password = ""  #Remove the password
```

Test connection Wifi

exemple016

```
import time
import network #Import network module

#Enter correct router name and password
ssidRouter     = 'XXXXXXXXXXXXX' #Enter the router name
passwordRouter = 'xxxxxxxxxxxxx' #Enter the router password

def STA_Setup(ssidRouter,passwordRouter):
    print("Setup start")
    sta_if = network.WLAN(network.STA_IF) #Set ESP32 in Station mode
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
```

```
#Activate ESP32’s Station mode, initiate a connection request to the
router
#and enter the password to connect.
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
#Wait for ESP32 to connect to router until they connect to each other
successfully.
        while not sta_if.isconnected():
            pass
#Print the IP address assigned to ESP32 in “Shell”.
    print('Connected, IP address:', sta_if.ifconfig())
    print("Setup End")

try:
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
```

Test LCD

exemple017

```
from time import sleep_ms, ticks_ms
from machine import I2C, Pin
from i2c_lcd import I2cLcd

DEFAULT_I2C_ADDR = 0x27

i2c = I2C(scl=Pin(22), sda=Pin(21), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)

lcd.move_to(1, 0)
lcd.putstr('Bonjour')
lcd.move_to(1, 1)
lcd.putstr('Au Castellab')

# The following line of code should be tested
# using the REPL:

# 1. To print a string to the LCD:
#    lcd.putstr('Hello world')
# 2. To clear the display:
#lcd.clear()
# 3. To control the cursor position:
# lcd.move_to(2, 1)
# 4. To show the cursor:
# lcd.show_cursor()
# 5. To hide the cursor:
```

```
#lcd.hide_cursor()
# 6. To set the cursor to blink:
#lcd.blink_cursor_on()
# 7. To stop the cursor on blinking:
#lcd.blink_cursor_off()
# 8. To hide the currently displayed character:
#lcd.display_off()
# 9. To show the currently hidden character:
#lcd.display_on()
# 10. To turn off the backlight:
#lcd.backlight_off()
# 11. To turn ON the backlight:
#lcd.backlight_on()
# 12. To print a single character:
#lcd.putchar('x')
# 13. To print a custom character:
#happy_face = bytearray([0x00, 0x0A, 0x00, 0x04, 0x00, 0x11, 0x0E,
0x00])
#lcd.custom_char(0, happy_face)
#lcd.putchar(chr(0))
```