Activités - Scripts micropython

Activité 2 : sortie numérique - commander une LED

Script initial

```
from machine import Pin
import time

pin_led = Pin(26, mode=Pin.OUT)

while True:

    pin_led.on()
    time.sleep(1)
    pin_led.off()
    time.sleep(1)
```

Modification :

```
from machine import Pin
import time

pin_led = Pin(26, mode=Pin.OUT)

while True:

    pin_led.on()
    time.sleep(2)
    pin_led.off()
    time.sleep(0.5)
```

Activité 3 : entrée numérique - lire un bouton

Script initial

```
from machine import Pin
import time

pin_led = Pin(26, mode=Pin.OUT)

while True:
  pin_bouton = Pin(25, mode=Pin.IN)
  if pin_bouton.value() == 1:
    pin_led.on()
    time.sleep(1)
  else:
    pin_led.off()
    time.sleep(1)
```

Modification

```
from machine import Pin
import time

pin_led = Pin(26, mode=Pin.OUT)

while True:
  pin_bouton = Pin(25, mode=Pin.IN)
  if pin_bouton.value() == 1:
    pin_led.off()
    time.sleep(1)
  else:
    pin_led.on()
    time.sleep(1)
```

Activité 4 : lire une entrée analogique - potentiomètre

Script initial

```
from machine import *
from time import *

def pinADC(pinNumber, db=ADC.ATTN_11DB, bit=ADC.WIDTH_10BIT):
  pin = ADC(Pin(pinNumber))
  pin.atten(db)
  pin.width(bit)
  return pin

# Potentiometer on p35
a2 = pinADC(35)

while True:
  print(a2.read())
  sleep(0.5)
```

Activité 5 : sortie PWM et LED

Script initial

```
from machine import *
from time import *

# LED on p26
pwm_led = PWM(Pin(26), freq=50, duty=0)

while True:
  for duty in range(0,1024, 50):
    pwm_led.duty(duty)
    sleep(0.2)
```

Modification :

```
from machine import *
from time import *

# LED on p26
pwm_led = PWM(Pin(26), freq=50, duty=0)

while True:
  for duty in range(1024, -1, -50):
    pwm_led.duty(duty)
    sleep(0.2)
```

Activité 6 : potentiomètre et LED

Script initial

```
from machine import *
from time import *

def pinADC(pinNumber, db=ADC.ATTN_11DB, bit=ADC.WIDTH_10BIT):
  pin = ADC(Pin(pinNumber))
  pin.atten(db)
  pin.width(bit)
  return pin
```

```
# Potentiometer on p35
pot = pinADC(35)

# LED on p26
pwm_led = PWM(Pin(26), freq=50, duty=0)


while True:
  vPot = pot.read()
  duty = vPot
  pwm_led.duty(duty)
  sleep(0.2)
```

Modification

```
from machine import *
from time import *

def pinADC(pinNumber, db=ADC.ATTN_11DB, bit=ADC.WIDTH_10BIT):
  pin = ADC(Pin(pinNumber))
  pin.atten(db)
  pin.width(bit)
  return pin



# Potentiometer on p35
pot = pinADC(35)

# LED on p26
pwm_led = PWM(Pin(26), freq=50, duty=0)


while True:
  vPot = pot.read()
  duty = vPot
  pwm_led.duty(duty)
  sleep(0.2)
```

Activité 7 : sortie PWM et servomoteur

Script initial

```
from machine import *
from time import *

for _ in range(2):
  print(0)
  setServoAngle(d6, 0)
  sleep(2)

  print(150)
  setServoAngle(d6, 150)
  sleep(1)


while True:
  pass
```

Modification

```
from machine import *
from time import *

# Servo on p27
d6 = PWM(Pin(27), freq=50, duty=0)
```

```
def setServoAngle(pin, angle):
  if (angle >= 0 and angle <= 180):
    pin.duty(int(0.025*1023 + (angle*0.1*1023)/180))
  else:
    raise ValueError("Servomotor angle have to be set between 0 and 180")


for _ in range(2):
  print(0)
  setServoAngle(d6, 45)
  sleep(2)

  print(150)
  setServoAngle(d6, 90)
  sleep(1)


while True:
  pass
```

Activié 8 : potentiomètre et servomoteur

Script initial

```
from machine import *
from time import *



def pinADC(pinNumber, db=ADC.ATTN_11DB, bit=ADC.WIDTH_10BIT):
  pin = ADC(Pin(pinNumber))
  pin.atten(db)
  pin.width(bit)
  return pin

def setServoAngle(pin, angle):
  if (angle >= 0 and angle <= 180):
    pin.duty(int(0.025*1023 + (angle*0.1*1023)/180))
  else:
    raise ValueError("Servomotor angle have to be set between 0 and 180")

def map(x, x1, x2, y1, y2):
  return x * (y2 - y1) / (x2 - x1) + y1

# Potentiometer on p35
pot = pinADC(35)

# Servo on p27
servo = PWM(Pin(27), freq=50, duty=0)

while True:
  vPot = pot.read()
  angle = map(vPot, 0, 1023, 0, 180)
  print(vPot, angle)
  setServoAngle(servo, angle)
```

Modification

```
from machine import *
from time import *



def pinADC(pinNumber, db=ADC.ATTN_11DB, bit=ADC.WIDTH_10BIT):
```

```
  pin = ADC(Pin(pinNumber))
  pin.atten(db)
  pin.width(bit)
  return pin

def setServoAngle(pin, angle):
  if (angle >= 0 and angle <= 180):
    pin.duty(int(0.025*1023 + (angle*0.1*1023)/180))
  else:
    raise ValueError("Servomotor angle have to be set between 0 and 180")

def map(x, x1, x2, y1, y2):
  return x * (y2 - y1) / (x2 - x1) + y1

# Potentiometer on p35
pot = pinADC(35)

# Servo on p27
servo = PWM(Pin(27), freq=50, duty=0)

while True:
  vPot = pot.read()
  angle = map(vPot, 0, 1023, 45, 90)
  print(vPot, angle)
  setServoAngle(servo, angle)
```