Pontifícia Universidade Católica de Goiás
Escola Politécnica e de Artes
ELT1119 – Redes e Aplicações IoT
Avaliação – AV1 - SUB
Nome:_____**Data:**_____

Lista de Exercícios – PWM e Funções em MicroPython (ESP32)

Questões Objetivas

1. Em um projeto de controle de LEDs com a placa **ESP32**, foi utilizado **PWM** para variar a intensidade do brilho de um LED. O código foi implementado utilizando a classe PWM da biblioteca machine e uma função personalizada para controlar o duty cycle de acordo com a entrada de um sensor. Abaixo estão algumas proposições sobre o comportamento do código e a técnica de PWM:

Proposições:

• I. O PWM permite controlar a potência média aplicada ao LED, variando a razão cíclica de um sinal digital.
• II. O valor máximo de duty cycle no ESP32, usando o método duty(), é 1023 por padrão.
• III. A frequência do PWM não afeta diretamente o brilho do LED, apenas a rapidez com que o sinal PWM é alternado.
• IV. Um duty cycle de 50% resultará em metade da potência aplicada ao LED, o que pode ser interpretado como aproximadamente 50% de brilho.

É correto afirmar que:

a) Apenas as proposições I e II estão corretas.
b) Apenas as proposições I, II e IV estão corretas.
c) Apenas as proposições III e IV estão corretas.
d) Todas as proposições estão corretas.

2. Um código foi escrito para controlar o brilho de um LED usando PWM no ESP32. O objetivo é ajustar o brilho em diferentes níveis, utilizando uma função personalizada para definir o valor do duty cycle. O código é mostrado abaixo:

```
from machine import Pin, PWM

led = PWM(Pin(15), freq=1000)

def set_brightness(level):
  if 0 <= level <= 100:
    duty = int(level * 1023 / 100)
    led.duty(duty)
  else:
    print("Nível inválido")

set_brightness(50)
```

Proposições:

• I. A função set_brightness() ajusta corretamente o duty cycle, variando o brilho de 0% a 100%.
• II. O duty cycle é definido proporcionalmente ao valor de level, sendo que level = 50 ajusta o brilho para 50% do máximo.
• III. O código deve ser ajustado para que a frequência do PWM seja configurada a cada chamada da função set_brightness().

É correto afirmar que:

a) Apenas a proposição I está correta.
b) Apenas as proposições I e II estão corretas.

c) Apenas as proposições II e III estão corretas.
d) Todas as proposições estão corretas.

3. Considere o seguinte código que foi escrito para controlar um servo motor em diferentes ângulos usando PWM no ESP32:

```
from machine import Pin, PWM
from time import sleep

servo = PWM(Pin(15), freq=50)

def set_servo_angle(angle):
  if 0 <= angle <= 180:
    min_duty = 40   # 0 graus
    max_duty = 115  # 180 graus
    duty = int(min_duty + (max_duty - min_duty) * angle / 180)
    servo.duty(duty)
  else:
    print("Ângulo inválido")

set_servo_angle(90)
sleep(1)
set_servo_angle(180)
```

Proposições:
• I. O código permite controlar o servo motor em ângulos de 0 a 180 graus com base no valor do duty cycle.
• II. A frequência de 50 Hz é adequada para a maioria dos servos.
• III. O código precisa de modificações para que o servo retorne à posição inicial automaticamente após 5 segundos.

É correto afirmar que:
a) Apenas a proposição I está correta.
b) Apenas as proposições I e II estão corretas.
c) Apenas as proposições II e III estão corretas.
d) Todas as proposições estão corretas.

4. O código a seguir foi escrito para controlar o brilho de um LED com dois botões no ESP32. Um botão aumenta o brilho, o outro diminui.

```
from machine import Pin, PWM
from time import sleep

led = PWM(Pin(15), freq=1000)
button_up = Pin(14, Pin.IN, Pin.PULL_UP)
button_down = Pin(13, Pin.IN, Pin.PULL_UP)

brightness = 512

def adjust_brightness(delta):
  global brightness
  brightness = min(1023, max(0, brightness + delta))
  led.duty(brightness)
```

```
while True:
  if not button_up.value():
    adjust_brightness(50)
    sleep(0.1)
  if not button_down.value():
    adjust_brightness(-50)
    sleep(0.1)
```

Proposições:
• I. O código permite controlar o brilho do LED de 0% a 100% utilizando os botões.
• II. O valor do brilho pode ser ajustado em incrementos de 50 a cada vez que o botão é pressionado.
• III. O código está configurado para variar o brilho do LED em steps pequenos, o que pode dificultar a percepção das mudanças.
É correto afirmar que:

a) Apenas a proposição I está correta.
b) Apenas as proposições I e II estão corretas.
c) Apenas as proposições II e III estão corretas.
d) Todas as proposições estão corretas.

5. O código a seguir simula um semáforo com LEDs no ESP32, controlando a intensidade por PWM:

```
from machine import Pin, PWM
from time import sleep

red = PWM(Pin(15), freq=1000)
yellow = PWM(Pin(16), freq=1000)
green = PWM(Pin(17), freq=1000)

def control_lights():
  red.duty(1023)
  yellow.duty(0)
  green.duty(0)
  sleep(5)

  red.duty(0)
  yellow.duty(1023)
  green.duty(0)
  sleep(2)

  red.duty(0)
  yellow.duty(0)
  green.duty(1023)
  sleep(5)

while True:
  control_lights()
```

Proposições:
• I. O código controla corretamente a sequência de um semáforo.
• II. O tempo de ativação de cada cor pode ser ajustado alterando o valor de sleep().
• III. O uso de botões para interação com pedestres pode ser uma melhoria funcional ao sistema.

É correto afirmar que:
a) Apenas a proposição I está correta.
b) Apenas as proposições I e II estão corretas.
c) Apenas as proposições II e III estão corretas.
d) Todas as proposições estão corretas.

6. O código abaixo ajusta o brilho de dois LEDs com base em valores inseridos pelo usuário, de 0 a 100%.

```
from machine import Pin, PWM

led1 = PWM(Pin(15), freq=1000)
led2 = PWM(Pin(16), freq=1000)

def set_brightness(led, level):
  if 0 <= level <= 100:
    duty = int(level * 1023 / 100)
    led.duty(duty)
  else:
    print("Valor de brilho inválido. Deve estar entre 0 e 100.")

while True:
  try:
    level1 = int(input("Digite o brilho para o LED 1 (0-100): "))
    level2 = int(input("Digite o brilho para o LED 2 (0-100): "))
    set_brightness(led1, level1)
    set_brightness(led2, level2)
  except ValueError:
    print("Por favor, digite um número inteiro válido.")
```

Proposições:
• I. O código ajusta corretamente o brilho dos dois LEDs com base nos valores inseridos.
• II. A função set_brightness() converte o valor percentual para o valor compatível com o PWM do ESP32.
• III. O código original não possuía proteção contra entradas inválidas, mas a versão atual evita erros com try-except.

É correto afirmar que:
a) Apenas a proposição I está correta.
b) Apenas as proposições I e II estão corretas.
c) Apenas as proposições II e III estão corretas.
d) Todas as proposições estão corretas.