

```
In [88]: # Import needed libraries
import threading
import time
import asyncio
import random
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")
import pynq.lib.rgbled as rgbled
```

```
In [89]: btns = base.btns_gpio
led5 = rgbled.RGBLED(5)
```

```
In [90]: # Number of philosophers
NUM_PHILOSOPHERS = 5

# Fifth philosopher LED
def blink_LED5(delay):
    for i in range(10):
        led5.write(0x2)
        time.sleep(delay)
        led5.write(0x0)
        time.sleep(delay)

# Function will turn LED5 off
def blink_LED5_off():
    led5.write(0x0)

# Function will turn an LED off
def LED_off(index):
    base.leds[index].off()

# Function will toggle an LED5 on and off
def blink(delay, index):
    for i in range(10):
        base.leds[index].toggle()
        time.sleep(delay)
        base.leds[index].off()
        time.sleep(delay)

# Take care of the fifth philosopher LED
def blink_led(index, delay):
    if (NUM_PHILOSOPHERS - 1) == index:
        blink_LED5(delay)
    elif (NUM_PHILOSOPHERS - 2) >= index:
        blink(delay, index)

# Take care of the fifth philosopher LED
def led_off(index):
    if (NUM_PHILOSOPHERS - 1) == index:
        blink_LED5_off()
    else:
        LED_off(index)

# Function to turn all LEDs off
def all_off():
    led5.write(0x0)
```

```
base.leds[0].off()
base.leds[1].off()
base.leds[2].off()
base.leds[3].off()
```

```
In [91]: # Set up four leds for four philosopher
led = [base.leds[i] for i in range(NUM_PHILOSOPHERS-1)]

forks = []
# Create five shared forks or locks
forks = [threading.Lock() for i in range(NUM_PHILOSOPHERS)]

# Create states
state = ""
eating = []
napping = []
thinking = []
eating = ["eating" for i in range(NUM_PHILOSOPHERS)]
napping = ["napping" for i in range(NUM_PHILOSOPHERS)]
thinking = ["thinking" for i in range(NUM_PHILOSOPHERS)]

# Creating an event to monitor threads
stop_threads = threading.Event()
btn = [btns[i] for i in range(4)]

# Task to to kill all threads, if any button is pushed
def kill_switch():
    while not stop_threads.is_set():
        if any(btn[i].read() for i in range(4)):
            print("Threads killed\n")
            stop_threads.set()
            all_off()
            time.sleep(0.1)
```

```
In [87]: # PART A2.1
def philosopher(i):
    # Create the left for and right fork next to it circularly as index loops
    left_fork = forks[i]
    right_fork = forks[(i+1)%NUM_PHILOSOPHERS]
    first, second = sorted((left_fork, right_fork), key=id)

    # Run foorever unless buttons are pushed to stop threads
    while not stop_threads.is_set():

        if (i != NUM_PHILOSOPHERS - 1):

            print("Philosopher {} is napping".format(i))
            blink(0.04, i)
            #time.sleep(0.3)

            # Philosopher is sleeping, turn off LED
            print("Philosopher {} is thinking/waiting\n".format(i))
            LED_off(i)
            time.sleep(0.5)

            # check if philosopher has first and second fork
            with first:
                with second:
```

```
        print("Philosopher {} is eating\n".format(i))
        blink(0.02, i)
        time.sleep(0.5)

    print("Philosopher {} is done eating, now going to nap\n".format(i))
    time.sleep(0.2)

# This branch executes the fifth philosopher LED
    elif i == NUM_PHILOSOPHERS - 1:

        print("Philosopher {} is napping\n".format(i))
        blink_LED5(0.04)
        #time.sleep(0.3)

        print("Philosopher {} is thinking/waiting\n".format(i))
        blink_LED5_off()
        time.sleep(0.5)

        with first:
            with second:
                print("Philosopher {} is eating\n".format(i))
                blink_LED5(0.02)
                time.sleep(0.5)

        print("Philosopher {} is done eating, now going to nap\n".format(i))
        time.sleep(0.2)

# array to store threads
    threads = []

# Start the five philosopher threads
    for i in range(NUM_PHILOSOPHERS):
        t = threading.Thread(target=philosopher, args=(i,))
        threads.append(t)
        t.start()

# Create the thread to monitor all buttons
    btn_thread = threading.Thread(target=kill_switch)
    btn_thread.start()

    for t in threads:
        t.join()
        print(f"{t.name} joined")

# Join the button thread to all threads
    btn_thread.join()
```

Philosopher 0 is napping  
Philosopher 1 is napping  
Philosopher 2 is napping  
Philosopher 3 is napping  
Philosopher 4 is napping

Philosopher 0 is thinking/waiting  
Philosopher 1 is thinking/waiting  
Philosopher 2 is thinking/waiting  
Philosopher 3 is thinking/waiting  
Philosopher 4 is thinking/waiting

Philosopher 0 is eating  
Philosopher 3 is eating

Philosopher 0 is done eating, now going to nap  
Philosopher 3 is done eating, now going to nap

Philosopher 4 is eating

Philosopher 0 is napping  
Philosopher 1 is eating

Philosopher 3 is napping  
Philosopher 4 is done eating, now going to nap

Philosopher 0 is thinking/waiting  
Philosopher 3 is thinking/waiting

Philosopher 1 is done eating, now going to nap

Philosopher 4 is napping

Philosopher 1 is napping  
Philosopher 2 is eating

Philosopher 0 is eating

Philosopher 4 is thinking/waiting

Philosopher 1 is thinking/waiting

Philosopher 2 is done eating, now going to nap

Philosopher 3 is eating

Philosopher 0 is done eating, now going to nap

Philosopher 2 is napping  
Philosopher 0 is napping  
Philosopher 1 is eating

Philosopher 3 is done eating, now going to nap

Philosopher 4 is eating

Philosopher 2 is thinking/waiting

Philosopher 3 is napping

Philosopher 0 is thinking/waiting

Philosopher 1 is done eating, now going to nap

Philosopher 1 is napping

Philosopher 2 is eating

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 3 is thinking/waiting

Philosopher 4 is napping

Philosopher 1 is thinking/waiting

Philosopher 2 is done eating, now going to nap

Philosopher 3 is eating

Philosopher 2 is napping

Philosopher 0 is done eating, now going to nap

Philosopher 4 is thinking/waiting

Philosopher 0 is napping

Philosopher 1 is eating

Philosopher 3 is done eating, now going to nap

Philosopher 4 is eating

Philosopher 2 is thinking/waiting

Philosopher 3 is napping

Philosopher 0 is thinking/waiting

Philosopher 1 is done eating, now going to nap

Philosopher 1 is nappingPhilosopher 2 is eating

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 3 is thinking/waiting

Philosopher 4 is napping

Philosopher 1 is thinking/waiting

Philosopher 2 is done eating, now going to nap

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 3 is thinking/waiting

Philosopher 4 is napping

Philosopher 1 is thinking/waiting

Philosopher 2 is done eating, now going to nap

Philosopher 3 is eating

Philosopher 2 is napping

Philosopher 0 is done eating, now going to nap

Philosopher 4 is thinking/waiting

Philosopher 0 is napping

Philosopher 1 is eating

Threads killed

Philosopher 3 is done eating, now going to nap

Philosopher 4 is eating

Philosopher 2 is thinking/waiting

Philosopher 0 is thinking/waiting

Philosopher 1 is done eating, now going to nap

Philosopher 2 is eating

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 2 is done eating, now going to nap

Philosopher 0 is done eating, now going to nap

Thread-137 (philosopher) joined

Thread-138 (philosopher) joined

Thread-139 (philosopher) joined

Thread-140 (philosopher) joined

Thread-141 (philosopher) joined

```
In [92]: # PART A2.2:
def gen_random_int(a,b):
    nap = random.randint(a,b)
    eat = random.randint(a,b)

    #if (nap>eat) or (nap == eat):
    #    gen_random_int(a,b)
    #return nap/10, eat/10 else:
    #return nap/10, eat/10
```

```

# if nap is > than eat, swap, if equal, decrement nap
if nap>eat:
    nap, eat = eat, nap
elif nap==eat:
    nap = nap - 1
return nap*0.1, eat*0.1 # keep delay between 0.1 and 0.9

def philosopher(i):
    # Create the left for and right fork next to it circularly as index loops
    left_fork = forks[i]
    right_fork = forks[(i+1)%NUM_PHILOSOPHERS]
    first, second = sorted((left_fork, right_fork), key=id)

    # Run foorever unless buttons are pushed to stop threads
    while not stop_threads.is_set():
        #generate numbers between 1 and 9
        nap, eat = gen_random_int(2,6)
        if (i != NUM_PHILOSOPHERS - 1):

            print("Philosopher {} is napping".format(i))
            blink(eat, i) #pass in 'eat' for slower blinks since eat>nap time
                           # we want the LED to blink faster while eating and slower
                           # The larger value for slower blinks and smaller value for
                           # Apply the same to all parameter for blink methods
            #time.sleep(0.3)

            # Philosopher is sleeping, turn off LED
            print("Philosopher {} is thinking/waiting\n".format(i))
            LED_off(i)
            time.sleep(0.2)
            # check if philosopher has first and second fork
            with first:
                with second:
                    print("Philosopher {} is eating\n".format(i))
                    #blink(eat, i)
                    #time.sleep(0.2)
                    blink(nap, i) # Passing 'nap' for quicker blinks since nap<eat
                print("Philosopher {} is done eating, now going to nap\n".format(i))
                time.sleep(0.2)
            # This branch executes the fifth philosopher LED
        elif i == NUM_PHILOSOPHERS - 1:

            print("Philosopher {} is napping\n".format(i))
            blink_LED5(eat) ## Passing 'eat' for slower blinks since eat>nap
            #time.sleep(0.3)

            print("Philosopher {} is thinking/waiting\n".format(i))
            blink_LED5_off()
            time.sleep(0.2)

            with first:
                with second:
                    print("Philosopher {} is eating\n".format(i))
                    #blink_LED5(eat)
                    #time.sleep(0.2)
                    blink_LED5(nap) # Passing 'nap' for quicker blinks since nap<eat
                print("Philosopher {} is done eating, now going to nap\n".format(i))
                time.sleep(0.2)

```

```
# array to store threads
threads = []

# Start the five philosopher threads
for i in range(NUM_PHILOSOPHERS):
    t = threading.Thread(target=philosopher, args=(i,))
    threads.append(t)
    t.start()

# Create the thread to monitor all buttons
btn_thread = threading.Thread(target=kill_switch)
btn_thread.start()

# Join philosopher threads
for t in threads:
    t.join()
    print(f"{t.name} joined")

# Join the button thread to all threads
btn_thread.join()
```



Philosopher 0 is napping  
Philosopher 1 is napping  
Philosopher 2 is napping  
Philosopher 3 is napping  
Philosopher 4 is napping

Philosopher 3 is thinking/waiting

Philosopher 3 is eating

Philosopher 1 is thinking/waiting

Philosopher 2 is thinking/waiting

Philosopher 1 is eating

Philosopher 3 is done eating, now going to nap

Philosopher 3 is napping  
Philosopher 0 is thinking/waiting

Philosopher 4 is thinking/waiting

Philosopher 4 is eating

Philosopher 1 is done eating, now going to nap

Philosopher 2 is eating

Philosopher 1 is napping  
Philosopher 3 is thinking/waiting

Philosopher 4 is done eating, now going to nap  
Philosopher 0 is eating

Philosopher 2 is done eating, now going to nap

Philosopher 3 is eating

Philosopher 4 is napping

Philosopher 2 is napping  
Philosopher 1 is thinking/waiting

Philosopher 0 is done eating, now going to nap  
Philosopher 1 is eating

Philosopher 0 is napping  
Philosopher 3 is done eating, now going to nap

Philosopher 3 is napping  
Philosopher 1 is done eating, now going to nap

Philosopher 4 is thinking/waiting

Philosopher 1 is napping  
Philosopher 2 is thinking/waiting

Philosopher 4 is eating

Philosopher 2 is eating

Philosopher 2 is done eating, now going to nap

Philosopher 2 is napping

Philosopher 0 is thinking/waiting

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 4 is napping

Philosopher 3 is thinking/waiting

Philosopher 3 is eating

Philosopher 0 is done eating, now going to nap

Philosopher 0 is napping

Philosopher 1 is thinking/waiting

Philosopher 1 is eating

Philosopher 3 is done eating, now going to nap

Philosopher 3 is napping

Philosopher 2 is thinking/waiting

Philosopher 1 is done eating, now going to nap

Philosopher 2 is eating

Philosopher 4 is thinking/waiting

Philosopher 1 is napping

Philosopher 4 is eating

Philosopher 3 is thinking/waiting

Philosopher 0 is thinking/waiting

Philosopher 2 is done eating, now going to nap

Philosopher 2 is napping

Philosopher 4 is done eating, now going to nap

Philosopher 0 is eating

Philosopher 3 is eating

Philosopher 4 is napping

Philosopher 1 is thinking/waiting

Philosopher 3 is done eating, now going to nap

Philosopher 3 is napping