```
import machine
import dht
import time
import random

FAN_OFF = 0
FAN_LOW = 1
FAN_MEDIUM = 2
FAN_HIGH = 3

ENA = machine.PWM(machine.Pin(5), freq=1000)  # PWM on Enable pin (GPIO5)
IN1 = machine.Pin(4, machine.Pin.OUT)  # Motor driver IN1 (GPIO4)
IN2 = machine.Pin(18, machine.Pin.OUT)  # Motor driver IN2 (GPIO18)
dht_sensor = dht.DHT22(machine.Pin(22))  # DHT22 Sensor (GPIO22)

PIR_SENSOR = machine.Pin(23, machine.Pin.IN)  # PIR Sensor on GPIO23

learning_rate = 0.2
discount_factor = 0.9
epsilon = 0.2

Q_table = [[random.uniform(-1, 1) for _ in range(4)] for _ in range(3)]

power_usage = {FAN_OFF: 0, FAN_LOW: 10, FAN_MEDIUM: 20, FAN_HIGH: 35}

energy_saved = 0.0
time_interval = 2
```

```
NO_OCCUPANCY_TIMEOUT = 15  # Time in seconds before turning fan off
last_motion_time = time.time()

def calculate_heat_index(temp, humidity):
    """Calculates Heat Index (HI) for comfort analysis."""
    return temp + 0.1 * humidity + (temp * humidity) / 100

def get_state(temp, humidity):
    """Determines thermal comfort state."""
    heat_index = calculate_heat_index(temp, humidity)
    if heat_index < 27:
        return 0  # Cool
    elif 27 <= heat_index < 32:
        return 1  # Warm
    else:
        return 2  # Hot

def get_reward(temperature, action, prev_action):
    """Reward function for RL optimization."""
    if 24 <= temperature <= 27 and action > 0:
        return 10
    elif action == 0 and temperature > 28:
        return -10
    elif action != 0 and action == prev_action:
        return 5
    else:
        return -3
```

```
def get_action(temp, humidity):
    """Choose an action using Q-learning (ε-greedy policy)."""
    state = get_state(temp, humidity)
    if random.uniform(0, 1) < epsilon:
        return random.randint(0, 3)
    else:
        return Q_table[state].index(max(Q_table[state]))

def control_fan(action):
    """Controls the fan speed and direction using the L298N motor driver."""
    if action == FAN_OFF:
        ENA.duty(0)
        IN1.value(0)
        IN2.value(0)
    elif action == FAN_LOW:
        ENA.duty(256)
        IN1.value(1)
        IN2.value(0)
    elif action == FAN_MEDIUM:
        ENA.duty(512)
        IN1.value(1)
        IN2.value(0)
    elif action == FAN_HIGH:
        ENA.duty(1023)
        IN1.value(1)
        IN2.value(0)

def calculate_energy_saved(prev_power, new_power):
```

```
    """Calculates energy saved in Wh (only when power decreases)."""
    global energy_saved
    if new_power < prev_power:
        energy_saved += (prev_power - new_power) * (time_interval / 3600)

while True:
    try:
        print("Reading DHT22 sensor...")
        dht_sensor.measure()
        temperature = dht_sensor.temperature()
        humidity = dht_sensor.humidity()

        prev_action = FAN_OFF if 'action' not in locals() else action

        # Check for motion
        if PIR_SENSOR.value() == 1:
            last_motion_time = time.time()  # Reset timeout
            motion_detected = True
        else:
            motion_detected = False

        # If no motion for timeout period, turn fan off
        if motion_detected or (time.time() - last_motion_time < NO_OCCUPANCY_TIMEOUT):
            action = get_action(temperature, humidity)
        else:
            action = FAN_OFF  # No occupancy, turn off fan

        prev_power = power_usage.get(prev_action, 0)
```

```
        new_power = power_usage.get(action, 0)
        calculate_energy_saved(prev_power, new_power)

        control_fan(action)

        print(f"Temp: {temperature:.2f}°C | Humidity: {humidity:.1f}% | Heat Index: {calculate_heat_index(temperature, humidity):.2f}°C | "
              f"Fan Speed: {action} | Power: {new_power}W | Energy Saved: {energy_saved:.2f}Wh | Motion: {'Detected' if motion_detected else 'No Motion'}")

    except Exception as e:
        print("Error reading sensor:", e)

    time.sleep(time_interval)
```