Smart Home Energy Management

Environment:
The environment is a smart home system managing energy usage. The agent is a smart controller that optimizes energy usage for heating, cooling, appliances while balancing energy costs, etc.

Reward:
The agent is rewarded for minimizing energy costs while maintaining comfort levels in a predefined range. If the temperature or comfort metrics deviate, or energy costs exceed a target, a penalty is to be applied.

Error:
Error occurs when the smart controller takes actions that lead to discomfort (room temperature going below 18°C or above 26°C). The goal is to minimize these errors by maintaining optimal settings.

Actor:
The actor decides actions like adjusting thermostat settings, scheduling appliance usage, or optimizing lighting based on the current state and the policy to maximize rewards.

Critic:
The critic evaluates the current state of the smart home (temperature, appliance usage, energy consumption) and predicts the long-term impact of the chosen actions on rewards.

Actions:

1. Increase the thermostat temperature.
2. Decrease the thermostat temperature.
3. Turn on/off appliances.
4. Adjust lighting levels.

States:

1. Initial state: All systems are off, and the environment is at a default temperature and energy usage.
2. Continuous states: Represent changes in temperature, energy consumption, and comfort levels based on the agent's actions.

A3C Flow:

1. The system starts with a predefined initial state, such as a room temperature of 22°C, appliances off, and moderate energy usage.
2. The actor takes an action (increase the thermostat by 2°C).

3. The environment responds to the action, updating the state (for example, new temperature is 24°C, and energy usage increases by 10%).
4. The critic evaluates the new state and calculates the advantage based on the reward (moderate reward for maintaining comfort but a slight penalty for increased energy usage).
5. The actor adjusts its policy based on feedback from the critic to select better actions in following steps.
6. The process repeats, aiming to balance energy costs and comfort over multiple iterations.

Pseudo-code for A3C Implementation:

```
# Initialize environment, parameters
initialize_environment()
initialize_actor_critic_models()
set_initial_state()

while not done:
   # Get the current state of the environment
   state = get_current_state()

   # Actor selects an action based on the policy
   action = actor_model.predict(state)

   # Apply action and observe new state and reward
   new_state, reward, done = environment.step(action)

   # Critic evaluates the new state
   value = critic_model.evaluate(new_state)

   # Calculate the advantage
   advantage = reward + gamma * value - critic_model.evaluate(state)

   # Update actor and critic models
   actor_model.update(state, action, advantage)
   critic_model.update(state, value)

   # Transition to the new state
   state = new_state
```

This problem can be adapted to any smart system aligning with the reinforcement learning framework.