1.Creational Design Pattern

**Factory Method:** With a lot of types of sensors and devices, a factory method for each of them can help in the creation of these objects. This will help a lot because you can introduce new types of sensors/devices into the program without breaking existing client code.

2.Structural Design Pattern

**Composite:** A room can contain both sensors and devices, forming a hierarchical structure. The composite pattern can represent this structure with sensors and devices as leaf nodes and rooms as composite nodes. This will allow us to work more efficiently and simplify CRUD operations.

**Decorator:** In a smart home, there are a lot of automatic actions. For example, if a person goes near a door, a motion sensor can caption this and open the door, and at the same moment a lightbulb can start functioning. And there can be lots more situations that can appear, so a decorator pattern will be perfect for this because you can combine several behaviors by wrapping an object into multiple decorators.

3.Behavioral Design Pattern

**Iterator:** Given the hierarchical structure of our smart home, an iterator could help us in the search for a device/sensor. Now we should go and check in every room if there is a sensor/device with a specified name, but an iterator would significantly simplify this operation.

**Observer:** Sensors are the core of a smart home, based on the data they collect, one or more devices react. The observer pattern can be very useful in this operation because multiple devices can monitor the data that sensors are collecting and react to it. Also, you can introduce new subscriber classes without having to change the original code, so if a new device is added, it can easily monitor the other sensors.