Temperature control in a greenhouse

Your challenge is to implement a heating and cooling system for your greenhouse. Your heating system will compare the actual temperature to a set-point, and turn on the heat if it gets too cold. If it gets too hot, a servo motor will open the top lid to promote passive cooling.

1. Draw a state diagram that captures the following behavior:

   - Periodically (upon expiration of a timer), the temperature and setpoint are read and compared.
   - When the temperature drops below a minimum (heating) setpoint, the heater turns on.
   - When the temperature rises above the setpoint, the heater turns off.
   - When the temperature rises above a maximum (cooling) setpoint, the lid opens.
   - When the temperature falls below the setpoint, the lid closes.

   Use the state diagram to help you write your control code. Be sure to create and use classes to your advantage (e.g., make a class for the AD22100 sensor).

2. You will use the AD22100 temperature sensor for measuring both the inside and outside temperatures. You will want to mount both in ways that they won't be bumped around. The interior sensor will need to be soldered to a small protoboard, which can then be attached to the wall of the greenhouse.

3. You will use a 24V power supply to *independently* supply the heating circuit. You will be given 33Ω power resistors to arrange as you best can to produce maximum heat without exceeding 2A. **Do not make any connections between the 24V circuit and the Arduino system!** Not even ground. What device will you use to control the relatively large power for the heaters?

4. You will use a servo motor to open and close the lid. I will provide you with a rod of the right length; you will 3d print a mount for the servo and a connector for the rod to the lid.

5. Before you start, draw a complete schematic of your proposed system using the components you have used up to this point. **Label the connections between components with the voltage and current limits.** For example, The connection from the power supply to the Arduino would have something like, "$V < 12V$" (the Arduino's limits), and, "$I < 2A$" (the power supply's limits).Show the schematic to an instructor before you plug anything in (you may start building the circuit – just don't fire it up until someone looks it over).

6. Add code to print both the setpoints and the current temperature to the serial monitor. Set the timer so that the sensors are read every 5 seconds. You may *not* use `delay()`.

Hysteresis

The system you developed in the previous section still has a minor flaw, namely that noise in the temperature sensing can lead to the relay switching on and off rapidly. In a full-sized heating system (e.g., a residential furnace), such *short-cycling* can lead to unnecessary wear and tear on the system; in ours, it could theoretically lead to premature failure of the relay. It's only a $1 relay, but still, we'd like to avoid the behavior.

As described in the reading, one way to combat this is to use *hysteresis*, or memory, which allows us to change our behavior based on past events. For the system at hand, you will add a variable that raises or lowers the nominal set-point, depending on whether we've received a call for heat or cooling.

1. Edit your state diagram to include a hysteresis band.
2. Implement a 1C hysteresis band. Make the band "below" the set-points. That is, the heater should come on when the temperature drops to 1C below set-point, and turn off when set-point is reached. Be sure to update your state transition diagram, which will make the programming task almost trivial.