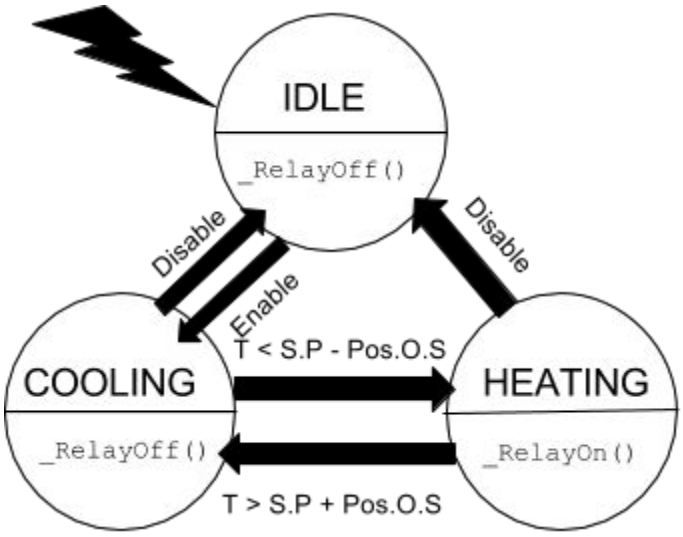
Sargis Yonan - Unit Tests & Edge Cases

Finite State Machine Logic and Modules in the Driver Libraries and main:

I tested the state machine logic by setting up and creating the hardware, compiling and burning the software onto the hardware, and testing the corresponding output with a given input. To test the logic, I used the temperature sensor with a hot-water bath and an ice bath. I placed the sensor in the ice-water bath, and the the LED attached to the controller board which represented a water heater's heating element, lit up. I then left the sensor probe in the hot water, representing the heating element heating up the water in a water heater. The LED stays lit until the temperature reaches a temperature over the deadband (setpoint + positive_offset), as expected. I then tried the reverse scenario (placing the probe back into the ice-bath), and once the temperature reached a temperature below the deadband (setpoint - negative_offset), the heating element (LED) turns back on. This indicated to me that the state machine I created functioned the way I intended it to. The program uses a finite state machine with pointers to functions to select the proper output function (_RelayOff() and _RelayOn()), but when the state machine has yet to receive an 'ENABLE' command from the wireless radio, the state machine stays in the _Idle() function which simply keeps the relay off because the system has not been enabled. The main function is responsible for interfacing the state machine to the sensors and relays. The final test for main included making sure every possible commands I created for the machine made the state machine transitions to the correct next state. I used print statements to debug the machine by making sure that a certain temperature range would force the FSM into the proper state with when enabled. main() never ends and the program is in an infinite loop because there is no OS for the microcontroller to return a value to from main, so the program is essentially a forever processing water heater until the power is turned off to the microcontroller. This functionality works, because I left the unit on for several days and attempted to remotely control it, and it worked as well as executed the proper feedback control for the heater. The working logic in the form of a state diagram is as follows:



XBEE Radio Module:

To test this piece of hardware, I connected the output and input pins of the radio to the RX and TX input pins corresponding to USART0 RX/TX ports on the microcontroller, and then gave the radio the proper voltage (+3.3V), reference voltage (+3.3V), and ground. I then configured the RX/TX on UART0 using a UART library I previously wrote for the microcontroller which allowed to configure the RX/TX ports to receive and send data synchronously with a baud rate (19200 in this case, but our tests found that transmissions worked between 9600 to 115200). I then used a serial communicator on my computer (CoolTerm for Mac OS X) with another XBEE radio attached acting as the master node in our star topology network. I then sent out various ASCII strings including, “hello, world!”, and they were printed back through the microcontroller which was connected to another channel of serial communication with CoolTerm configured with the correct baud rate. I then tested the machine’s abilities to receive commands from the master radio, using a function called ProcessCommand() in driver.c/h, that uses an Interrupt Service Routine to process an incoming request only when the receiving buffer of the microcontroller met 3 bytes (a full command). The program behavior works as expected, and by printing out the RX buffer, it is apparent that the hex commands are received, parsed, and processed. For example, I can send an ‘ENABLE’ command to the state machine via the master XBEE, and the microcontroller connected to the slave node, responded by waking up the state machine and turning the heating element on. I also found that the XBEE does not work at all with voltages below +2.9V, so it is important to use a separate power supply for the radio until a better one is found. The XBEE also does not work over +3.6V, this can also damage the radio, so a voltage regulator is important.