Objective: Basically what was said in your project proposal. Create a thermostat that closes a relay and turns on a fan if the temperature of the temperature gets 5 degrees above a set point or off if the temperature gets 5 degrees below the set point. Note the unit will need to be able to change the set point, by turning the knob, and should have a time readout. A really good system would have a different set points, for different times of the day.

Hardware Added: For this project an LM34, temperature-to-voltage device, is used measure the room temperature. A simple schematic, taken from the Texas Instrument datasheet, showing how this device is to be connected appears in Figure 1.

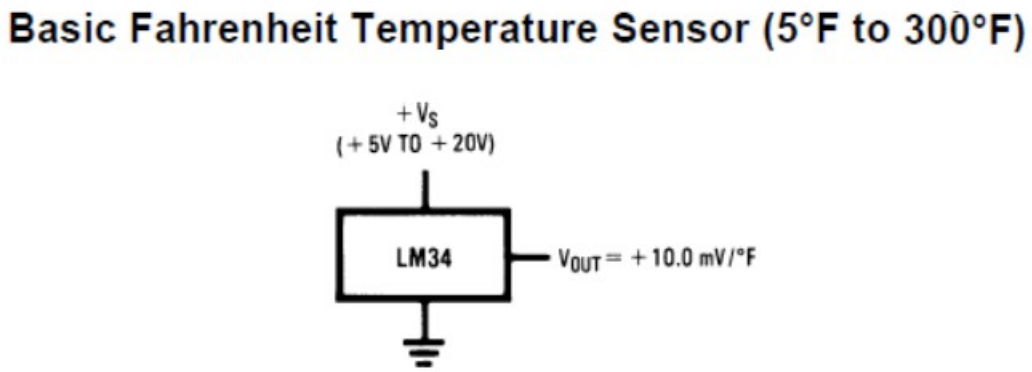

Figure 1. Basic Schematic of LM34 Circuit.

The Vout in figure 1 will be attached to A0 on the Arduino. In order to get reasonable precision on temperature the 3.3 volt from the Arduino will be used as a reference for the ADC. This means we will have a temperature range of 0 to 330 (3.3 / 10 mV) degrees Fahrenheit, much larger that we plan on using for a home thermostat.

The unit be turning on a fan to emulate a cooling system. The pins of the Arduino cannot supply the current needed for a motor, so a relay system was acquired that will switch on the power to the fan. The basic circuit of the board is shown in Figure 2. A separate board was chosen as an easy way of adding all the protection, optoisolation and flyback diode, with out having to wire all this up.

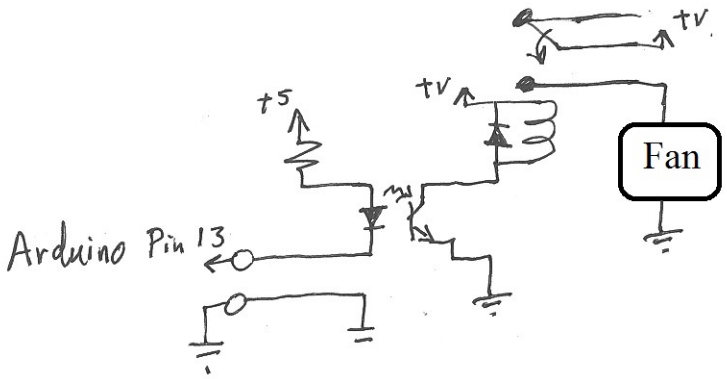

Figure 2. Relay Board Schematic

Software: In this section a documentation of the software design should be included. Documentation can include pseudo code and state transition diagrams.

The encoder knob will be as an input to the software, thus reading the A-B channels will be handled with interrupts, the code for which is in Appendix A. Also included in this code for debouncing of the push button, the state transition diagram of which is included in Figure 3.

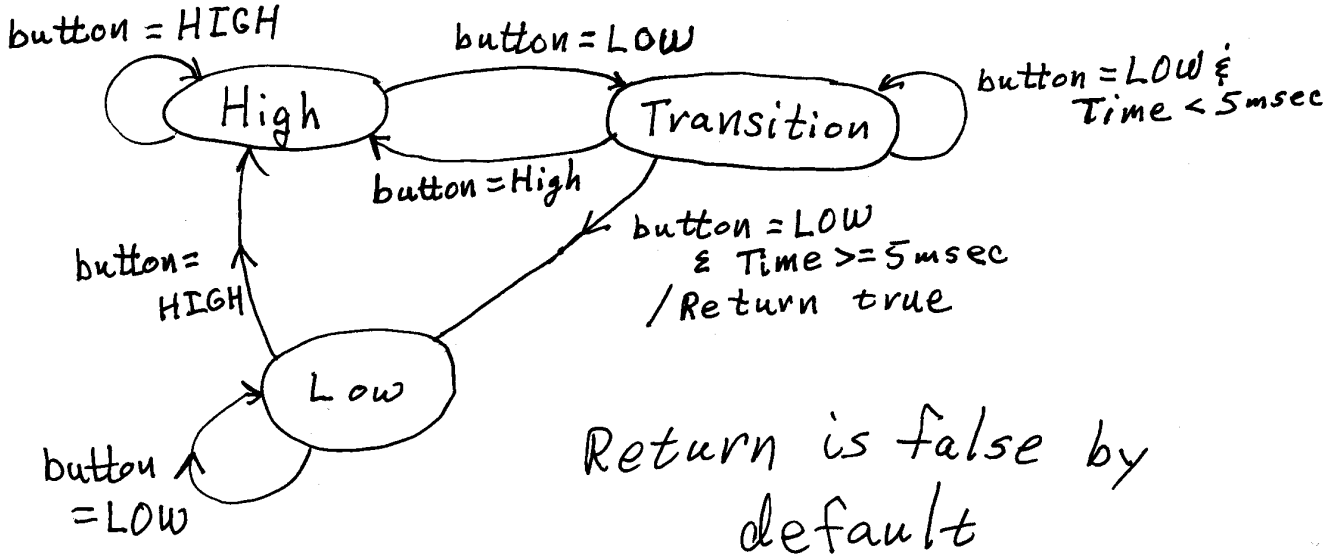

Figure 3. Push Button Debounce State Machine.

Note the code that is include needs to be formatted for easy reading.

The main code, or loop code, is rather simple and can be written is a simple block of pseudo code.

```
Global Variables:
   SetPointTemperature
   EncoderCurrentPosition

loop:
   if 1 second has passed,
       Read analog 0.
       Convert to temperature.
       if temperature is greater than the SetPointTemperature + Tolerance
           Turn on Fan
       else if temperature is less than the SetPointTemperature - Tolerance
           Turn off Fan
       while EncoderPostion - EncoderCurrentPosition > 4
            Increase temperature by 1 degree
            and add 4 to EncoderCurrentPosition
       while EncoderPostion - EncoderCurrentPosition < -4
            Decrease the temperature by 1 degree
            and subtract 4 from EncoderCurrentPosition
       Display temperature, SetPointTemperature and Fan state on LCD.
```

Testing: At this stage of development an elaborate test setup was not considered necessary, rather the simple heat source of human fingers were used to heat up the sensor, and the fan was observed as the temperature rises.

Appendix A: Code for Handling Inputs for the Encoder and Push Button.

```
// Setup for the button debounce State Machine.
enum buttonStates { High, Transition, Low };
int buttonState = High;
unsigned long buttonTimer;

// Function that will direct buttonState to the next state,
// and signal that the button has transitioned low.
int nextButtonState(int buttonInput) // buttonInput is from pin 4
{
	int ReturnValue = 0; // Defaults to false.
	// Based on State and input return next state.
	switch (buttonState)
	{
	default:
	case High:  // pin is high and waiting for low transition
		if (buttonInput == LOW)
		{
			buttonState = Transition;	// change to time out state
			buttonTimer = millis(); // record time of low transition
		}
		break;
	case Transition:
		if (buttonInput == HIGH)  // if pin goes high reset.
			buttonState = High;
		else if (millis() - buttonTimer >= 5) // if it has been low
		{                                     // for 5 millisec.
			buttonState = Low;    //Move to low,
			ReturnValue = true;     // and set the flag for event.
		}
		break;
	case Low:
		if (buttonInput == HIGH)  // if the pin goes high
			buttonState = High;    // go back to initial state.
		break;
	}// End of State switch

	return ReturnValue;
}// End of nextButtonState
```

```
// Variable for keeping track of encoder change.
volatile int EncoderValue = 0;
// Service Routine for Encoder line A
void PinA(void)
{
	// compare pin 3	to	pin 2
	if ((PIND >> 3 & 0x01) == (PIND >> 2 & 0x01))
	{
		EncoderValue++;
	}
	else
	{
		EncoderValue--;
	}
}// End of EncoderMonitor PinB

// Service Routine for Encoder line B
void PinB(void)
{
	if ((PIND >> 3 & 0x01) != (PIND >> 2 & 0x01))
	{
		EncoderValue++;
	}
	else
	{
		EncoderValue--;
	}
} // End of EncoderMonitor PinB
```

Appendix B: Code for Part 2.

```
// If nothing else make the code a different font to identify it
// and make sure it is properly spaced for easy reading.
void setup()
{
  // put your setup code here, to run once:
}

void loop()
{
  // put your main code here, to run repeatedly:

}
```