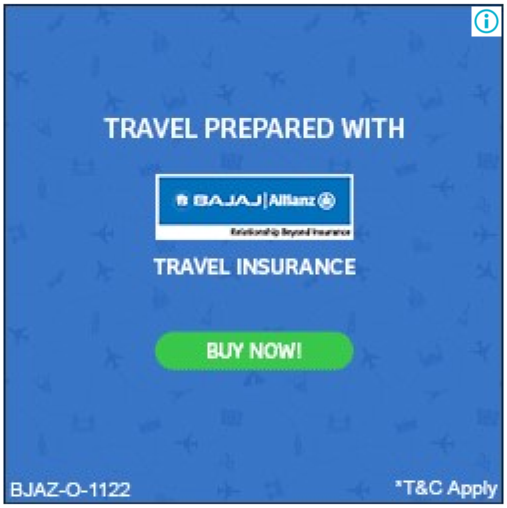
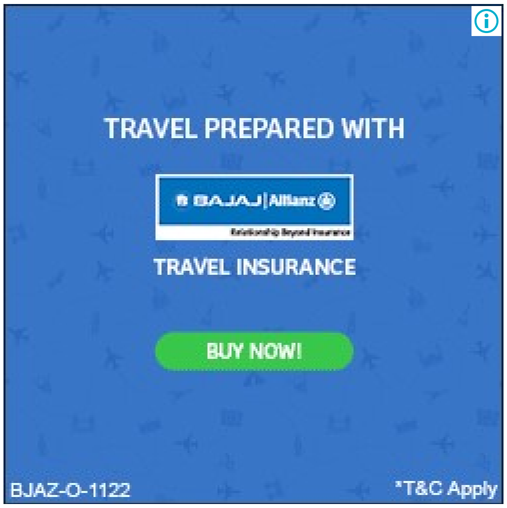


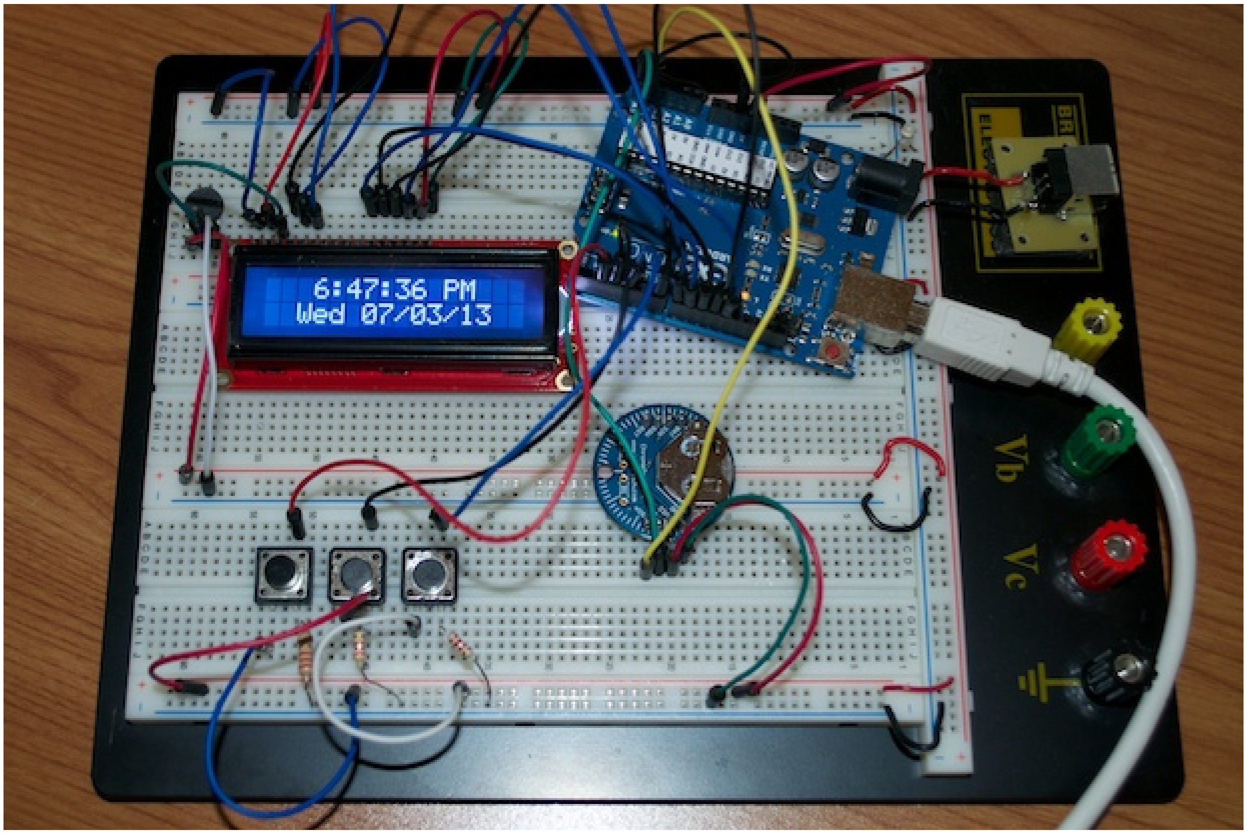


Reading the Temperature from the DS3231 & DS18B20

Jul 04, 2013 Filed in: Electronics | Arduino | Clocks

One thing that I definitely want to add to Rev 2 of my LCD clock is the ability to display the temperature. I found out that since the DS3231 has a temperature compensated internal crystal, you can poll it for the temperature. I wondered how accurate it would be. What good is displaying the DS3231's temperature if it runs 10 degrees over ambient? There was only one way to find out.

I quickly threw together a clock on my breadboard so I could play with the code.

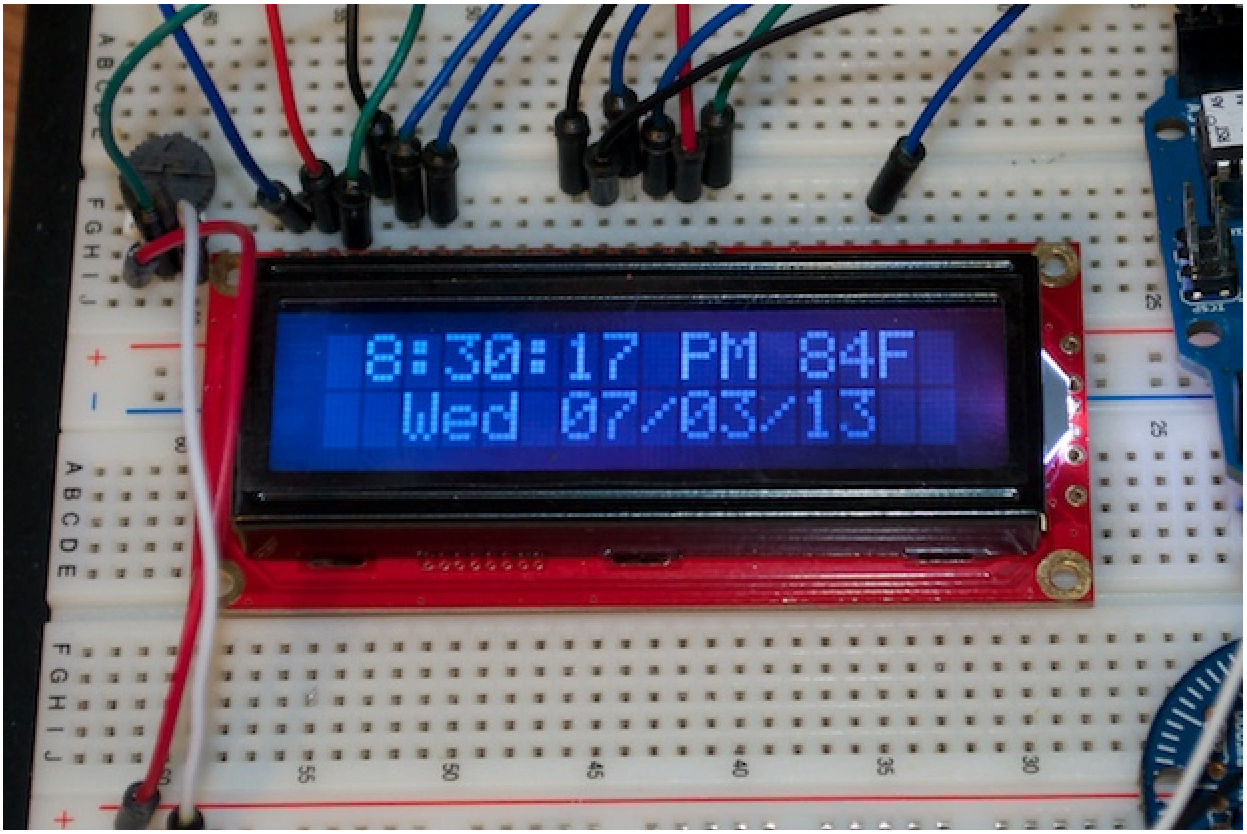


I then needed to interleave the following code into my clock sketch to read in and display the temperature:

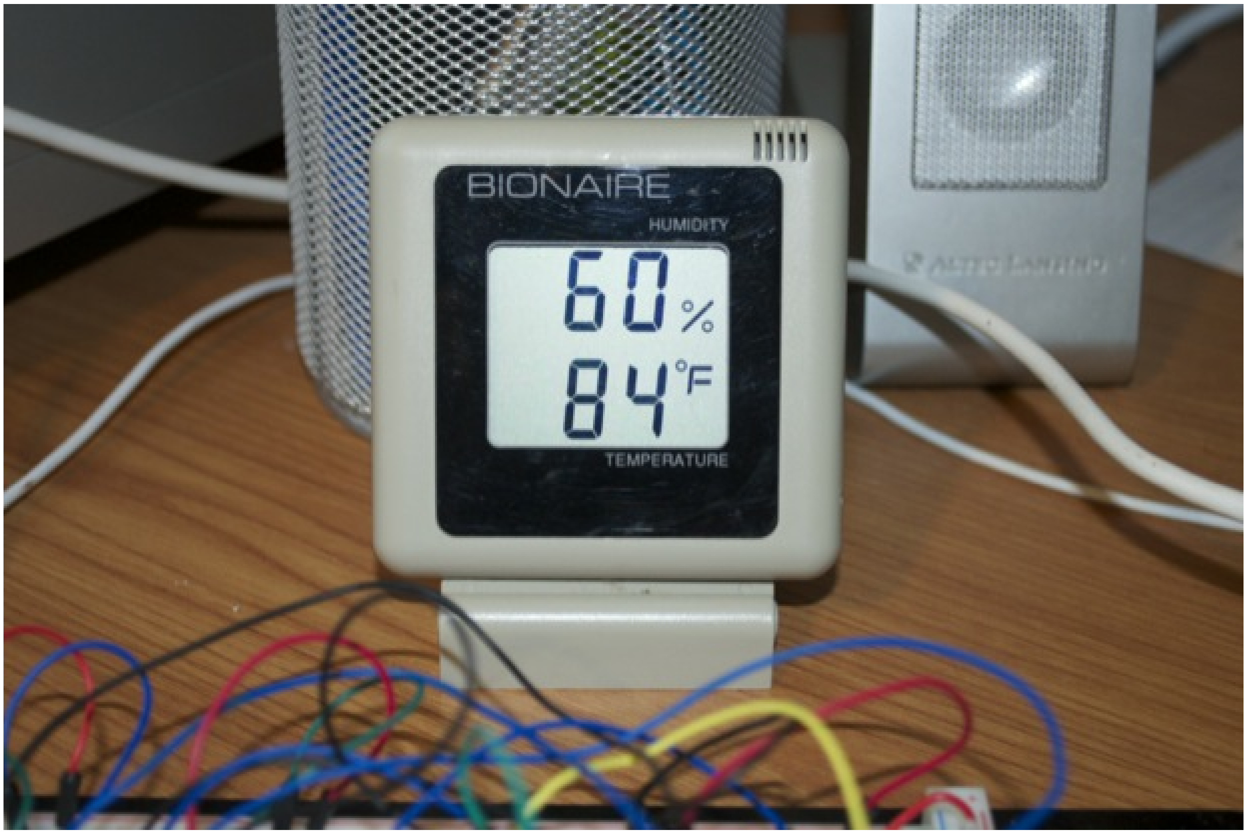
```
#include "DS3231.h"

DS3231 RTC;
int tempC;
int tempF;

void setup() {
  RTC.begin();
}

void loop() {
  tempC = RTC.getTemperature();
  tempF = (tempC * 1.8) + 32.0; // Convert C to F

  lcd.print(tempF);
```

```
    lcd.print("F");
  }
```

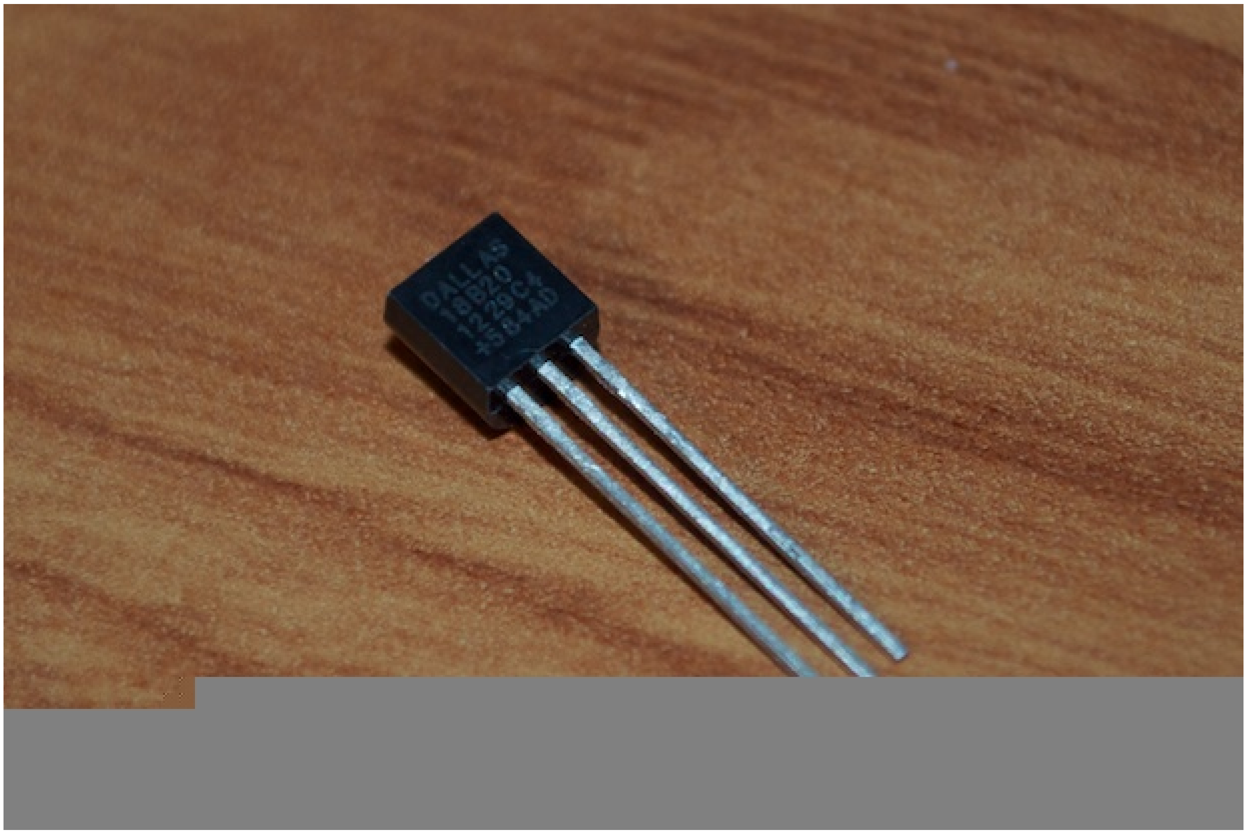
That was pretty easy.



I know it’s pretty hot in the computer room, but is it really 84 degrees? I grabbed my Bionaire thermometer to compare temperatures. Yup. It’s hot. That’s why I’ve been sitting next to the fan until my wife agrees that it’s hot enough to put on the AC.

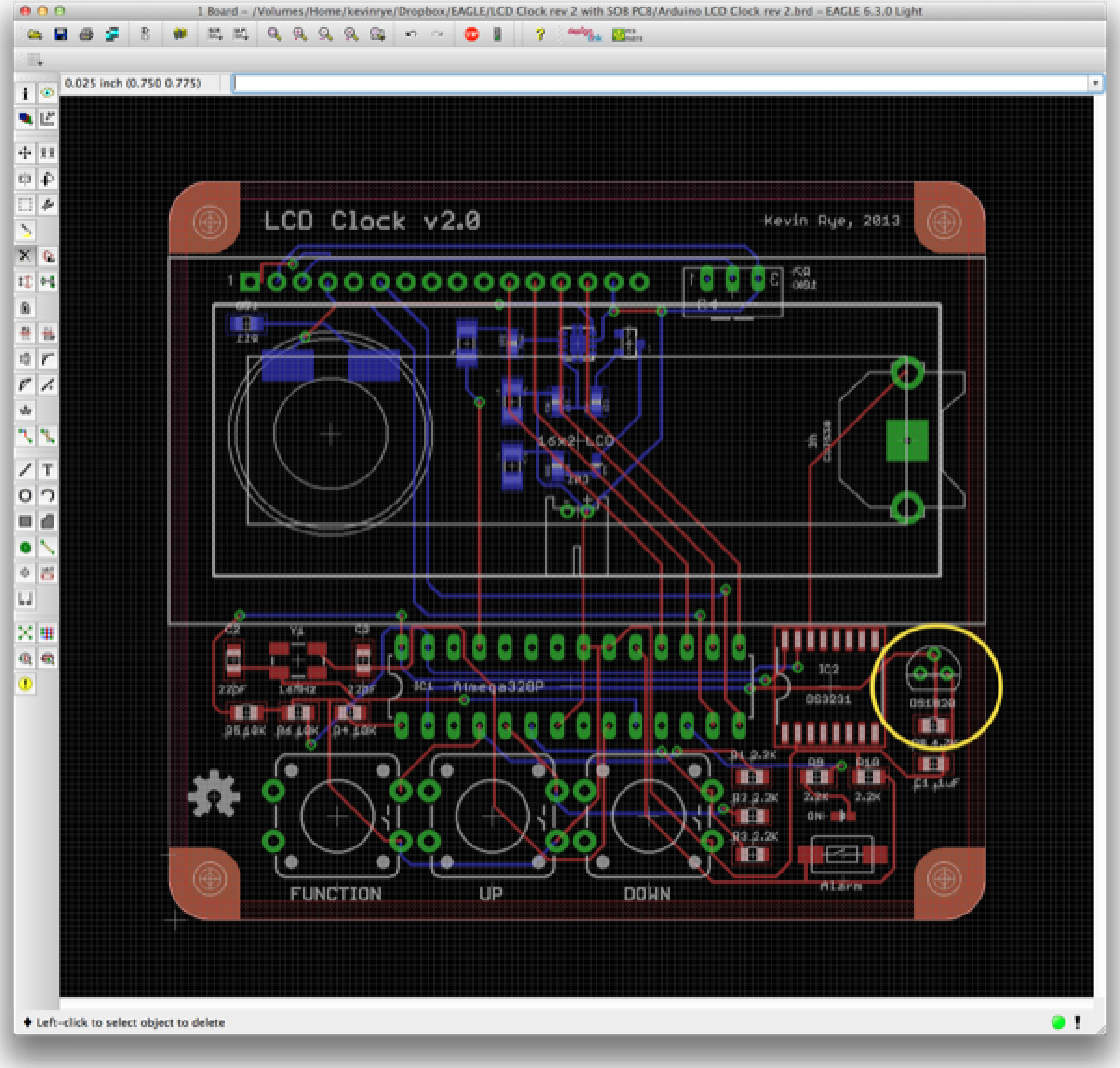


I picked up a DS18B20 temperature sensor a few months ago to play around with.



It's a pretty sweet sensor. It uses the OneWire library. Just like the name suggests, you only need one pin to talk to it. The cool thing about it is that each DS18B20 comes with a unique identifier. So you can have as many sensors on the bus as you want and you'd been able to pick out any single sensor and read in the temperature. That would be pretty handy in picking out hot spots in a large assembly.

I knew it was most likely going to find its way into Ver 2 of my LCD clock, so I went ahead and incorporated it into the PCB. (Even though I hadn't had the chance to play with it yet.) For Ver 2 of the clock, I changed all the parts to SMDs, so there was plenty space to add it.



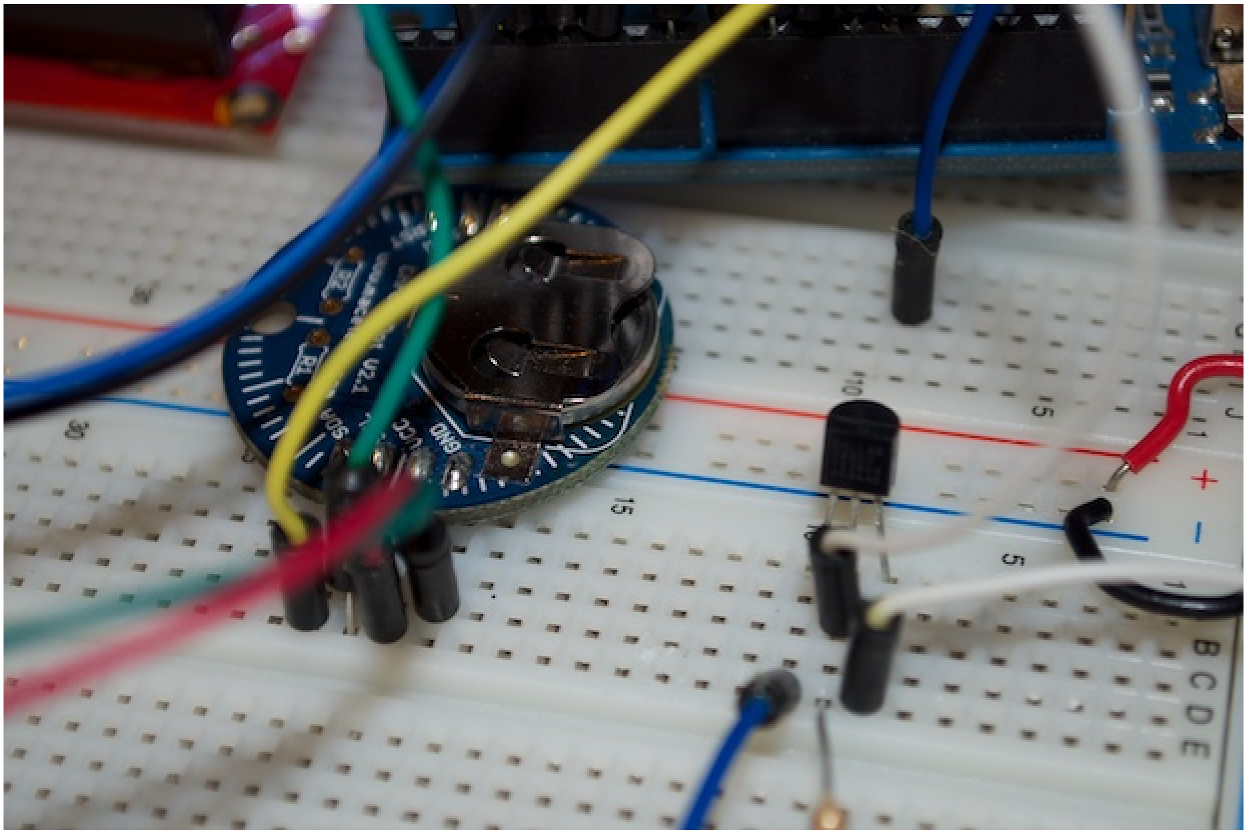
1 Board - /Volumes/Home/kevinrye/Dropbox/EAGLE/LCD Clock rev 2 with SO8 PCB/Arduino LCD Clock rev 2.brd - EAGLE 6.3.0 Light

0.025 inch (0.750 0.775)

Left-click to select object to delete

However, seeing that the DS3231 does a pretty good job of displaying the ambient temperature, I wondered if it even made sense to leave the DS18B20 in the design.

To compare the temperatures, I placed the DS3231 and the DS18B20 together on the breadboard.

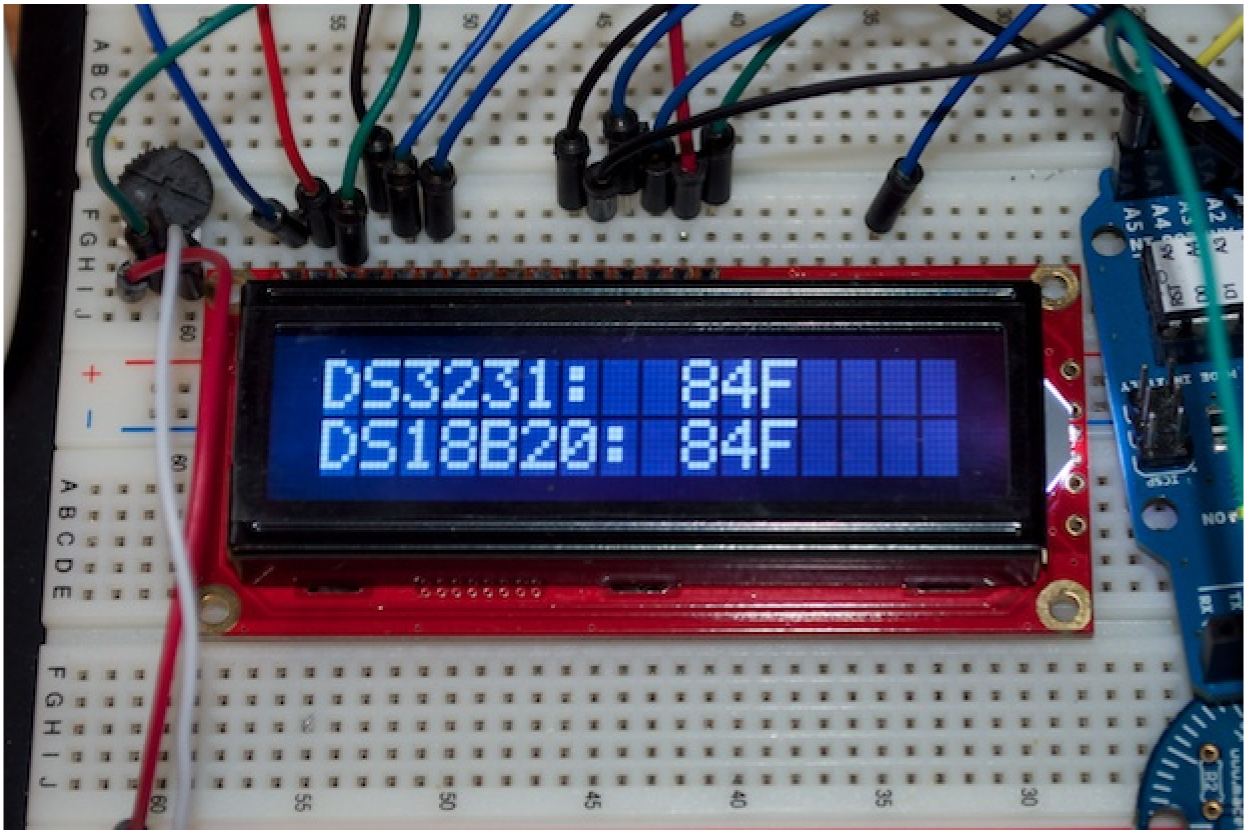


I then linked to the OneWre library and dropped in the code example posted on Bildr.org to talk to the DS18B20.

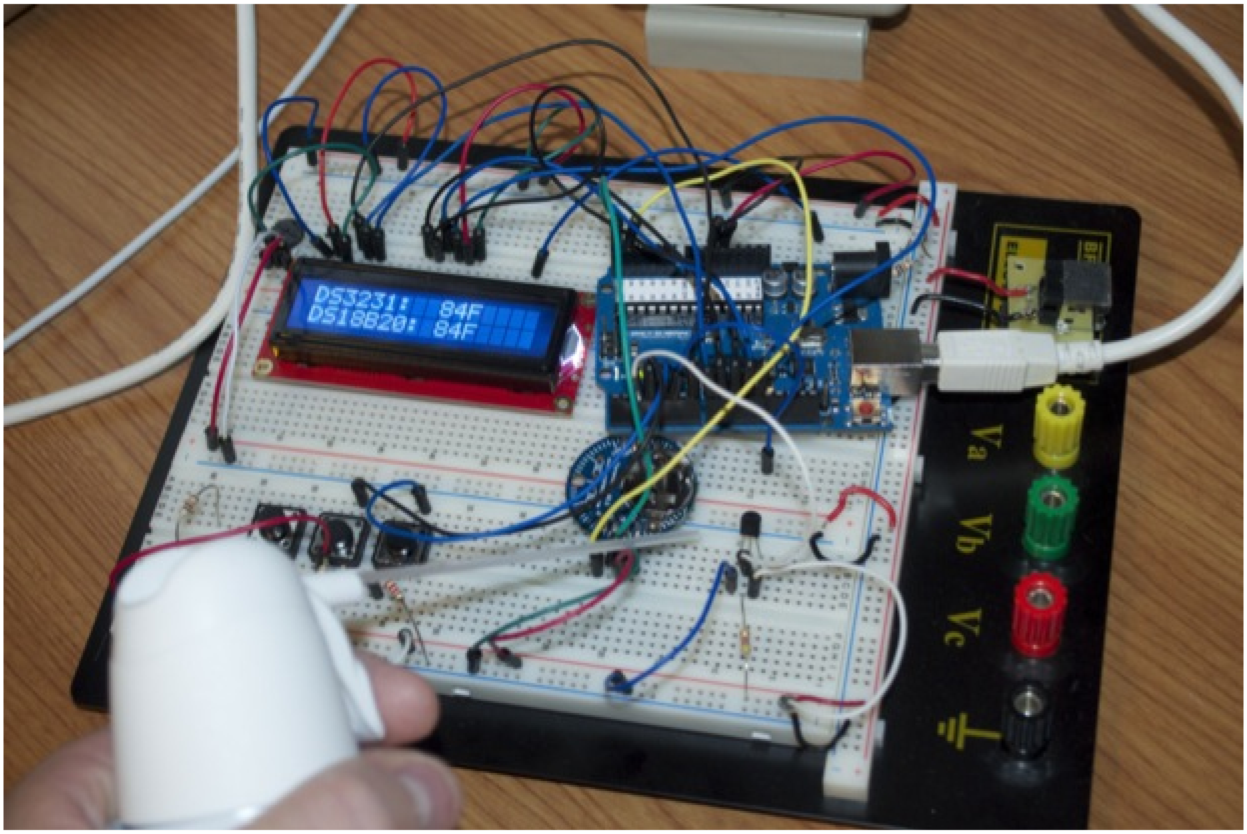
Here’s the complete sketch with the DS3231 and DS18B20 code:

```
#include <Wire.h>
#include <OneWire.h>
#include "DS3231.h"
#include <LiquidCrystal.h>

DS3231 RTC; //Create the DS3231 object

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

//temp
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2
OneWire ds(DS18S20_Pin);  // on digital pin 2
int tempCDS3231;
int tempFDS3231;
int tempCDS18B20;
int tempFDS18B20;

void setup() {

  Serial.begin(9600);
  Wire.begin();
  lcd.begin(16, 2);
  RTC.begin();
}

void loop() {

  //DS3231
  tempCDS3231 = RTC.getTemperature();
  tempFDS3231 = (tempCDS3231 * 1.8) + 32.0; // Convert C to F

  //DS18B20
  tempCDS18B20 = getTemp();
  tempFDS18B20 = (tempCDS18B20 * 1.8) + 32.0; // Convert C to F

  lcd.clear(); // clear LCD screen
  lcd.setCursor(0,0);

  //DS3231 temp
  lcd.print("DS3231:  ");
  lcd.print(tempFDS3231);
  lcd.print("F");

  //DS18B20 temp
  lcd.setCursor(0,1);
  lcd.print("DS18B20: ");
  lcd.print(tempFDS18B20);
  lcd.print("F");

}

//DS18B20
float getTemp() {
  //returns the temperature from one DS18S20 in DEG Celsius

  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Device is not recognized");
    return -1000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1); // start conversion

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad


  for (int i = 0; i < 9; i++) { //need 9 bytes
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB); //using two's compliment
  float TemperatureSum = tempRead / 16;

  return TemperatureSum;
}
```

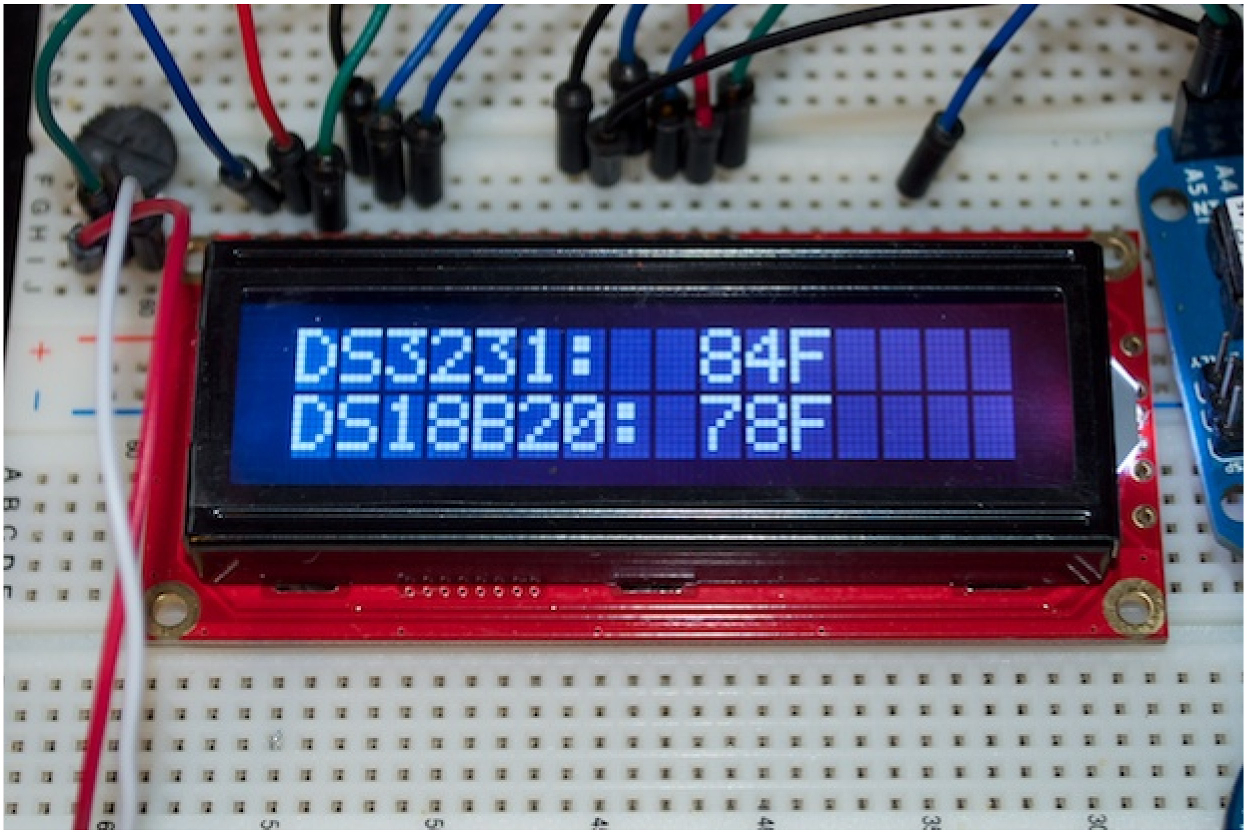
Cool! They both work. Both are reading exactly 84 degrees.



Just to make sure I was getting 2 separate readings, I hit the DS18B20 with some canned air.



The temperature immediately dropped to 78 degrees. It works.

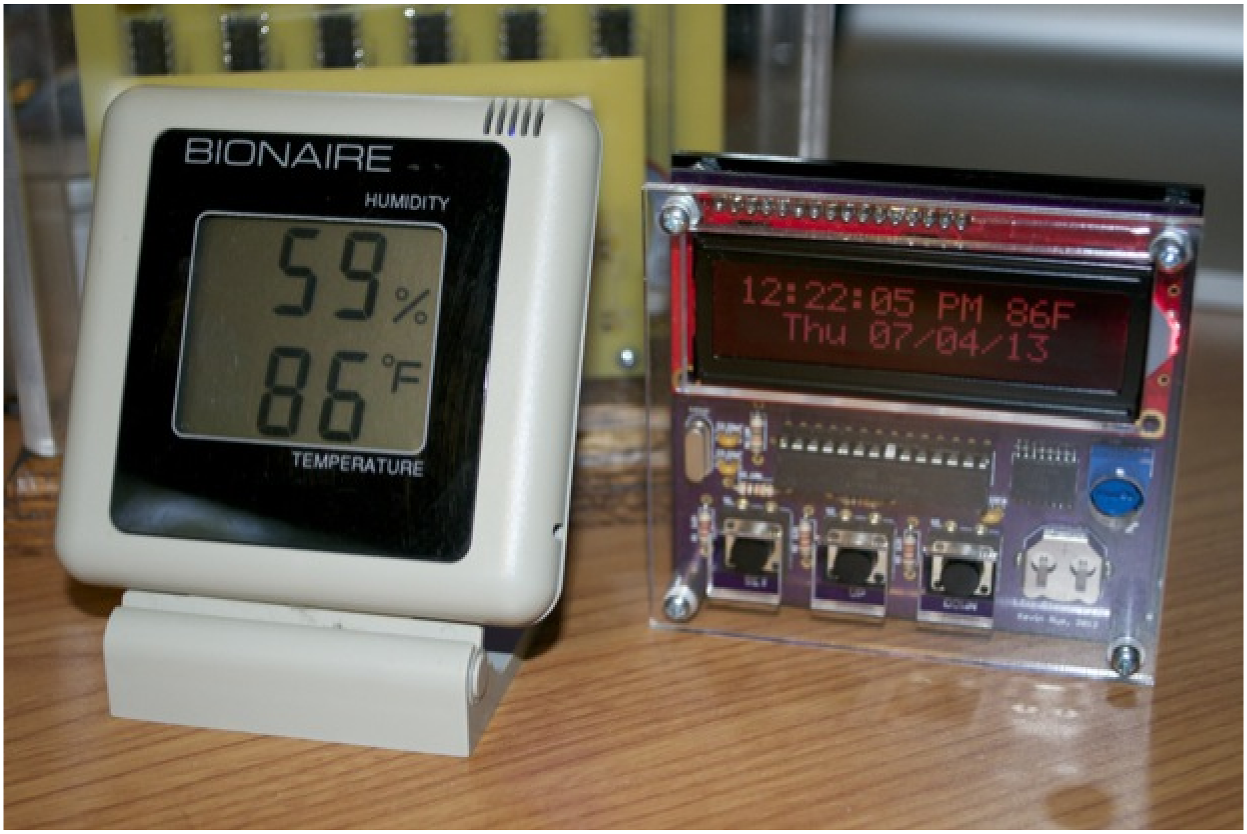


I let them run for a few hours while periodically comparing temperatures. I could have done something a little more "scientific" by logging temperatures, but that seemed to be a little overkill. What I did notice is that the DS3231 seems to be more stable. The DS18B20 jumps around a bit. It'll jump from 84 degrees to 82 and then back to 84 here and there.

That was last night. When I woke up this morning, both sensors were reading 80 degrees. The DS18B20 toggled between 80 and 82 a few times, but eventually settled at 80 degrees, just like the DS3231. They both eventually crept back up to 84 by mid day. I'm polling the temperature every second, so I could always change it to something like, say, once per minute to get a better average.

In any case, it looks like I won't need the DS18B20 anyway. The temperature from the DS3231 is more than adequate. But…yes, there's a but….will the DS3231 display the correct temperature when it's sitting behind a sheet of acrylic? Does the fact that it's "closed up" interfere with airflow/heat dissipation?

Time for one more test. I uploaded the final sketch to my Ver 1 clock and monitored the temperature for a few hours. It looks like it's pretty stable. The DS3231 consistently displayed the same temperature as the Bionaire. We have a winner!



- Main >

© 2006 - 2017 Kevin Rye. All Rights Reserved.