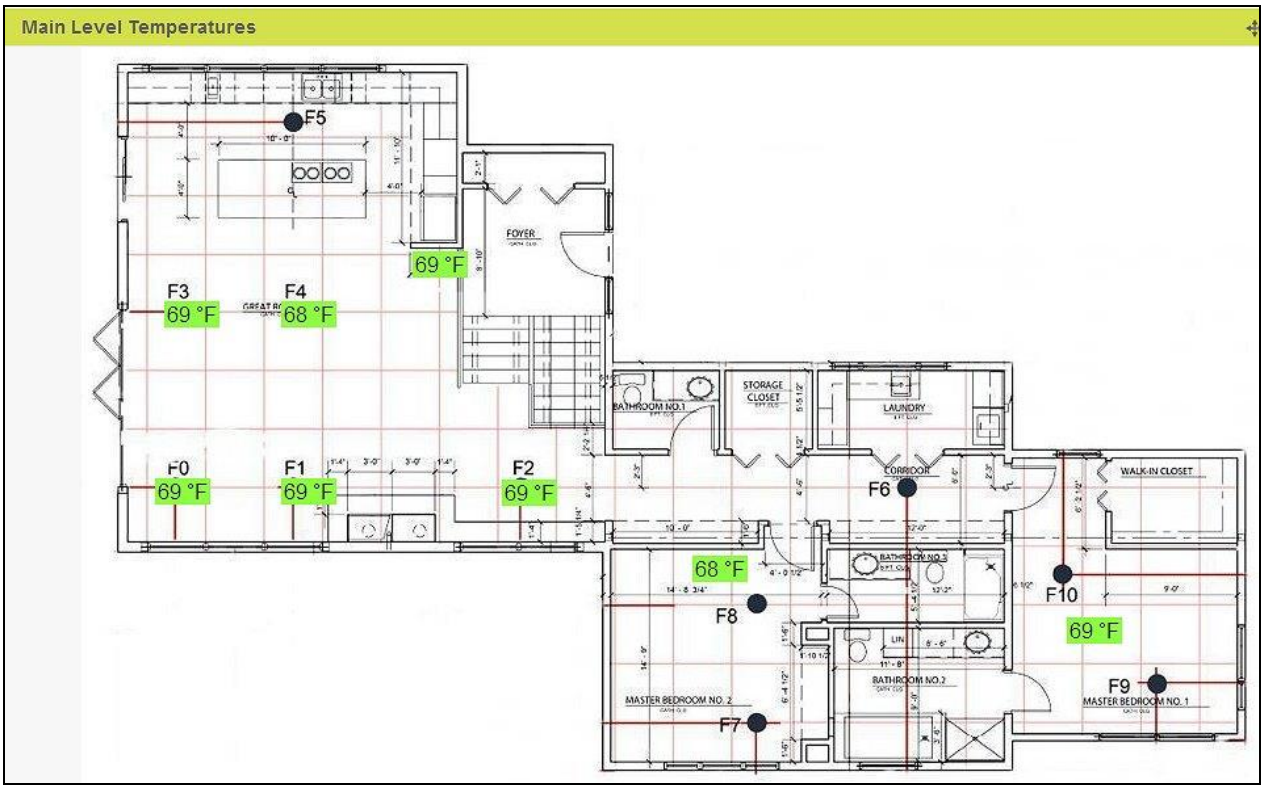
Household Temperature Monitoring System



By Mark Gagner and Phil Gagner

NeoTerra
DESIGN - BUILD - LIVE

1 Introduction

We have recently completed construction of a passive solar home in the beautiful Appalachian foothills of North Georgia. We call the project (and the house) *neoTerra*.

A passive solar design relies on south-facing glazing to collect heat from the sun during daylight and thermal mass to retain the heat and release it slowly in the evening. A true passive solar design will have glazing equal to about 15% (or more) of the floor space and it requires substantial thermal mass to moderate the temperature. This thermal mass is generally provided by a concrete slab that receives direct sunlight.

Our house has a single level of living space with a full, unfinished basement. Glazing on the main level is approximately 15% of the floor space so we constructed the house with a suspended concrete slab for the main level to provide the required thermal mass. Details on construction of the suspended slab are outside the scope of this note and can be found on our web site at http://www.neoterra.us/1/category/suspended%20slab/1.html.

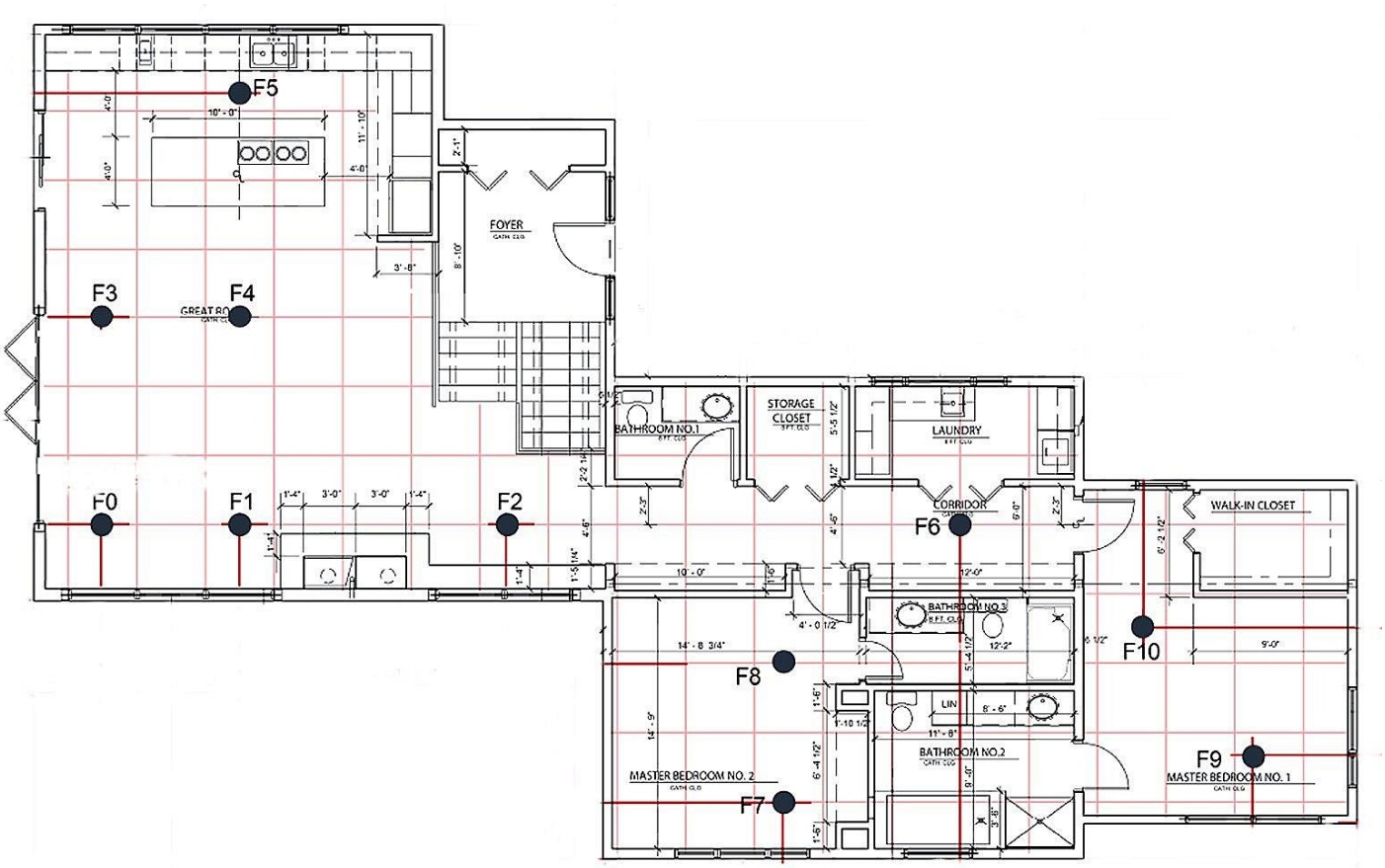
The floors are exposed concrete scored on a four foot grid.

Exposed Concrete Floor

There's a lot of engineering info on how much heat can be stored in various forms of thermal mass but there isn't very much practical info on how long it takes for a slab to warm up in the sun, how warm does it actually get under typical conditions and how fast does it cool off at night.

When were designing *neoTerra* we decided to incorporate a temperature monitoring system so we could continuously collect detailed temperature information throughout the structure. To get an understanding of how the thermal mass behaves under different conditions we placed five sensors in the basement slab (on grade), eleven sensors in the main level suspended slab and four sensors in the main level ceilings above the drywall.

The sensors have been positioned so that some are in areas that receive direct sun, some receive partial sun and a few that receive no sun. This will allow us to collect real life data that can be shared with other passive solar enthusiasts.



Main Level Sensor Placement

In the future, we will add sensors for outside air and soil temperature, as well as inside air temperature sensors.

2 System Design

My brother Phil is a software engineer so he took up the challenge of designing and building a low cost temperature monitoring system which will allow us to collect detailed data on how the passive solar design is functioning. This chapter describes the design and the components that were selected. A list of suppliers is provided at the end of the document.

2.1 Design Objectives

The design of the temperature monitoring system was based on the following criteria:

- Temperature sensors may be placed anywhere within the structure of the home: including concrete, walls, and ceilings. In addition, sensors may be placed outside the home.
- The sensors must be able to be strung along a run of wire in a multi-drop configuration.
- The wiring interface must be reliable, robust, and easily installed.
- The sensors may be as far as two hundred (200) feet from the host controller.
- The sensors must be durable in order to survive extreme environments.
- The controller must be automated.
- The controller must be reliable and inexpensive.
- Controller software should be freely available and based on a “common” language.
- A PC will be used to collect data from the controller for display as charts or graphs.

2.2 Hardware

There are four main elements in the system:

- Temperature sensors
- A controller that periodically retrieves measurements from the sensors
- An interface board for the sensors
- A small PC that is used to archive the data and publish data readings to the web

These elements are described in the following sections.

2.2.1 Temperature Sensors

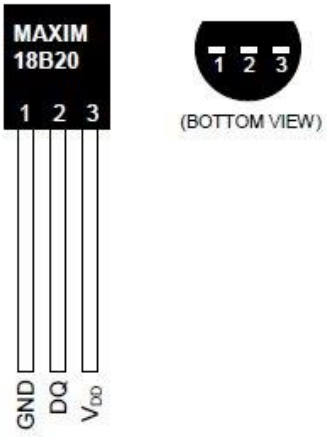
The temperature sensors are the most critical aspect of the project. Since many of them would be buried in concrete or exposed to the elements, we wanted a device that’s rugged and durable as well as accurate.

We selected the Maxim DS18B20 *1-Wire* digital thermometer. This device is capable of measurements between -55C and +125C, plus/minus 0.5C so it has a range and accuracy sufficient for any solar project.

In addition, this is a digital sensor that measures temperature and reports the value over a serial data bus. This means that the accuracy is not affected by the distance of the cable run. Maxim has tested the DS18B20 to remote distances of three hundred (300) meters with excellent results. Cabling will be described in the section on installation.

The DS18B20 supports a full multi-drop configuration whereby many sensors can be attached along a single cable run similar to a ladder with rungs. This reduces the cost and labor for installation of a system with a lot of sensors distributed over a large area.

The 'raw' sensor comes in a TO92, 3-pin package similar to a standard transistor. As depicted below.



The DS18B20 has only three pins:

- **DQ:** A bi-directional open-drain data pin supporting a proprietary serial communication protocol referred to as ***1-Wire***. The terms DQ and 1-Wire will be used interchangeably within this application note.
- **VDD:** This is the 5 volt power pin
- **GND:** Ground

The standard package is probably suitable for protected, interior locations. However, we wanted a more robust package that could be embedded in the concrete slabs or installed in harsh exterior locations. Fortunately, harsh environment versions of the sensor are available from several online vendors. These probes are encased in a stainless steel sleeve and have a heavy-duty cable.

DS18B20 Harsh Environment Temperature Probe

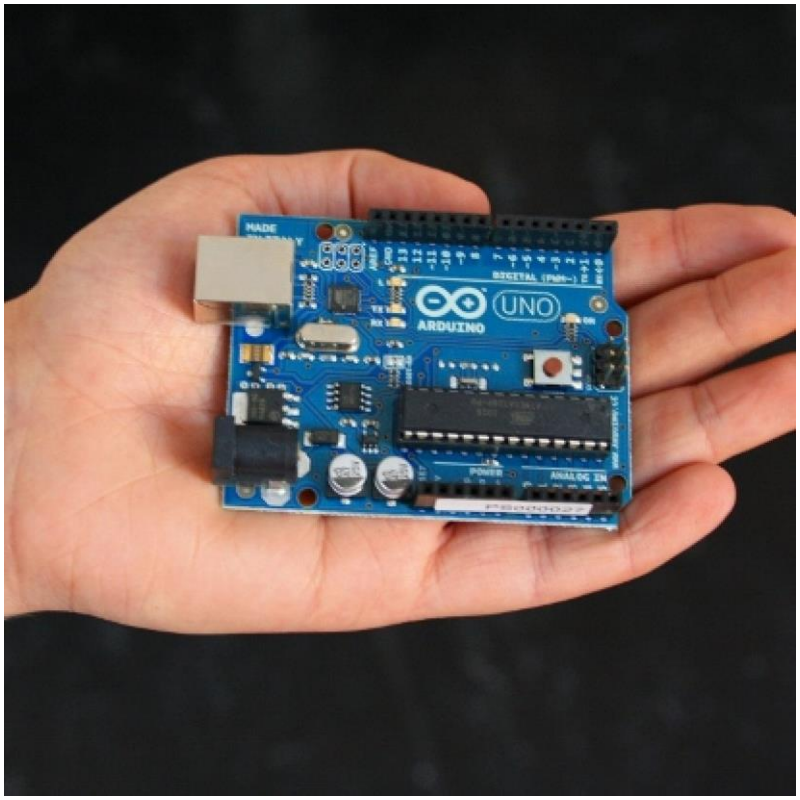
2.2.2 Controller

A small controller is required to communicate with the sensors and collect the measurements. The controller for this project is an Arduino Uno (Revision 3) open source micro-controller board based on the Atmega328 chip. The Arduino was selected for its low cost and because it has extensive support among hobbyists with numerous books and websites devoted to Arduino programming.

Revision 3 of the Uno supplies the following features:

- 14 digital I/O pins
- 6 analog I/O pins
- 16 MHz ceramic resonator
- USB connection
- DC power jack
- 3 LEDs: Tx, Rx, and Power
- 32 Kbytes Flash Memory
- 2 Kbytes Static RAM
- 1 Kbytes EEROM

The Uno, illustrated in the picture below, is fairly limited in its abilities. Even so, only a small percentage of the available functions are used in this project so we have spare capability to monitor other types of sensors in the future.



Arduino Uno Controller

Specifically, the Uno uses the following features in this design:

- USB port: This port has two purposes
  - Serial communications to the PC. This is programmed at 9600 baud.
  - Provides power to the Ardino Uno
- I2C serial protocol

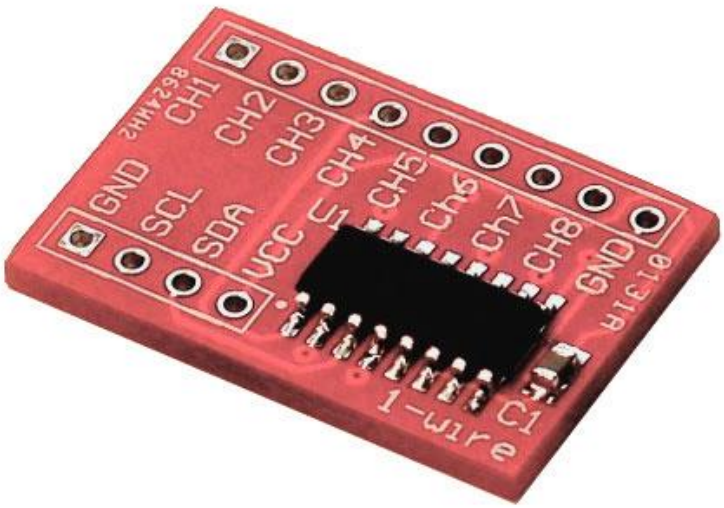
2.2.3 1-Wire Shield

The Uno supports the ***I2C*** serial protocol but the temperature sensors utilize the ***1-Wire*** protocol. There is a software library available for the Arduino to implement a single *1-Wire* channel but we wanted to have separate channels for each floor.

In the Arduino world, the word ***shield*** is used to describe hardware that interfaces to the micro-controller, generally as a piggy-back via headers.

Shields are designed by third-party vendors to provide features not resident on the Arduino itself. Many of the interface shields are quite simple and could be constructed by a serious hobbyist.

We used a third-party I2C to *1-Wire* interface board from InMojo.com as the communications liaison between the Arduino controller and the temperature sensors. The board is little more than a Maxim DS2482-800 interface chip mounted on a small PCB as illustrated below:

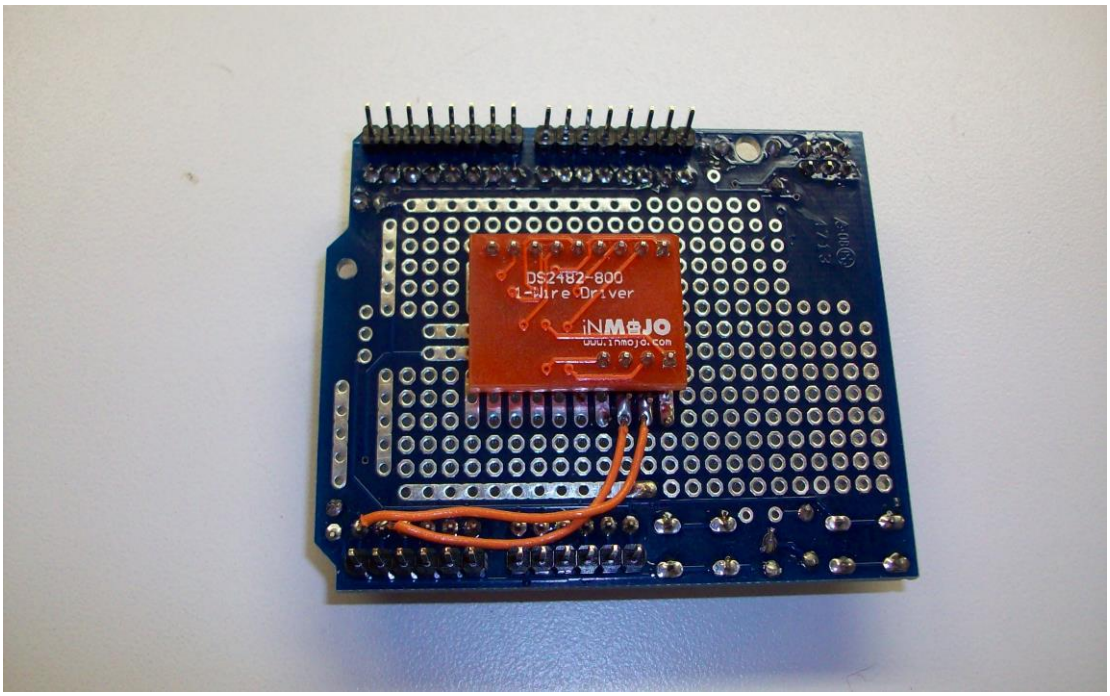


I2C to *1-Wire* Interface PCB

This is an ideal solution since it provides eight (8) *1-Wire* channels. We are using four in the initial installation which leaves four more for future expansion.

The InMojo board does not come in the form factor of a standard Arduino shield so we mounted it on a development/prototyping shield from Adafruit, a popular Arduino hobbyist site. The *1-Wire* PCB along with two (2) 1.8k ohm pull-up resistors are soldered to the Adafruit development shield and it, in turn is piggy-backed to the Arduino

The final assembly is shown in the picture below. Note that the *1-Wire* PCB is the red component and it is attached to the bottom of the development shield.



***1-Wire* Shield Assembly**

As mentioned, the Arduino supports I2C directly and could support a single *1-Wire* channel. The most significant advantage of the *1-Wire* shield is that the DS2482-800 supports eight (8) channels and provides the proper *1-Wire* timing specifications for the sensors independent of the I2C timing generated by the Arduino controller. This simplifies the Arduino coding and provides a more reliable solution.

Eight (8) independent *1-Wire* channels allow an almost unlimited number of sensors. The initial project utilizes four of the eight channels.

- Channel 0 - Basement floor temperature sensors
- Channel 1 – Main level floor temperature sensors
- Channel 2 – Embedded ceiling sensors
- Channel 3 – Interior and exterior air temperature sensors

Note that all of these sensors could have been driven from a single *1-Wire* channel. The main purpose of breaking them into independent logical groups was for reliability and future expandability.

2.2.4 PC

The Arduino has very limited memory and display capabilities so a small PC is used to display and archive data. It also publishes data to the web so it can be viewed remotely.

Just about any small PC could be used for this function. Since this PC will run 24/7 for the next several years we selected the Intel DN2800MT fan-less motherboard with a solid-state disk (SSD) for reliability and low cost.

Intel DN2800MT Motherboard

The PC connects directly to the house Ethernet and connects to the Ardunio controller via a USB cable configured as COM3. The USB provides bi-directional communications (set at 9600 baud) and DC power for Arduino. In turn, the Arduino supplies power for the *1-Wire* shield and all of the *1-Wire* sensors.

2.3 Completed System

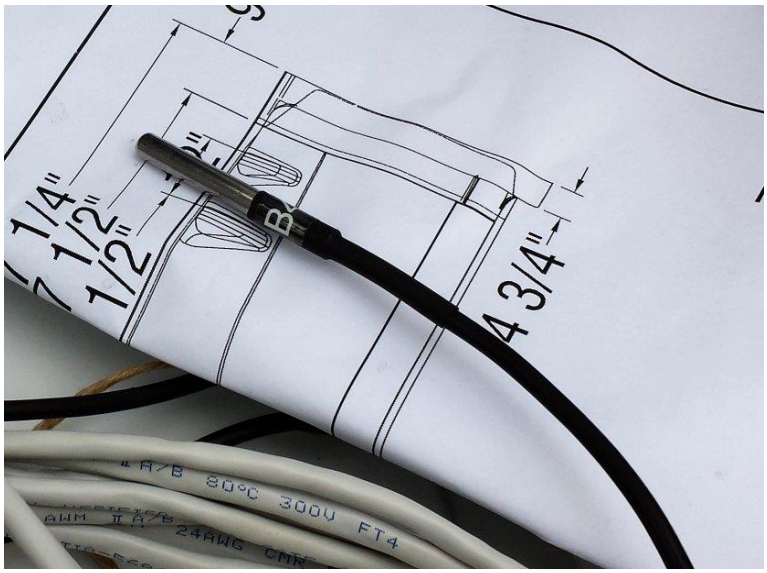
The complete system consists of the PC (on the right) and the Arduino+shield (on the left)

PC and Arduino Controller

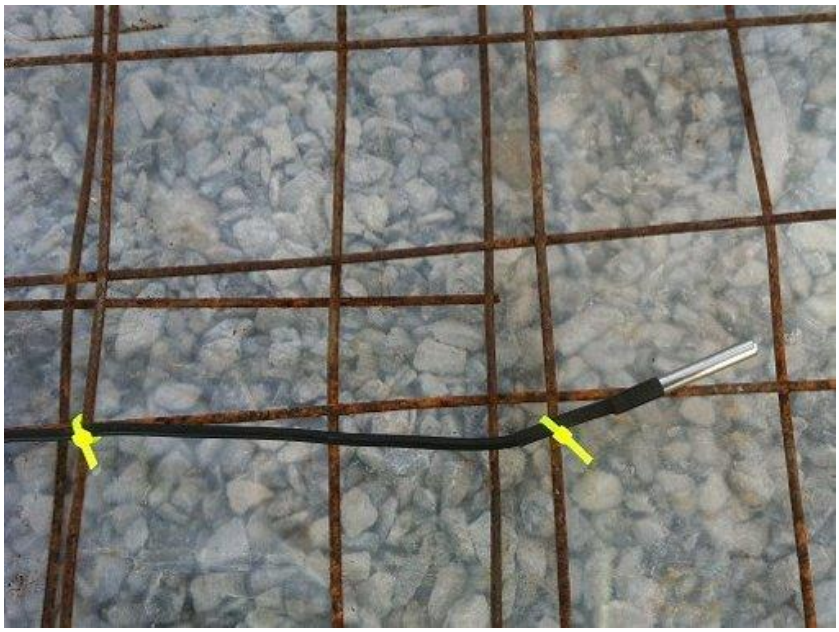
3 Installation

The temperature sensors are very rugged so installation was fairly simple.

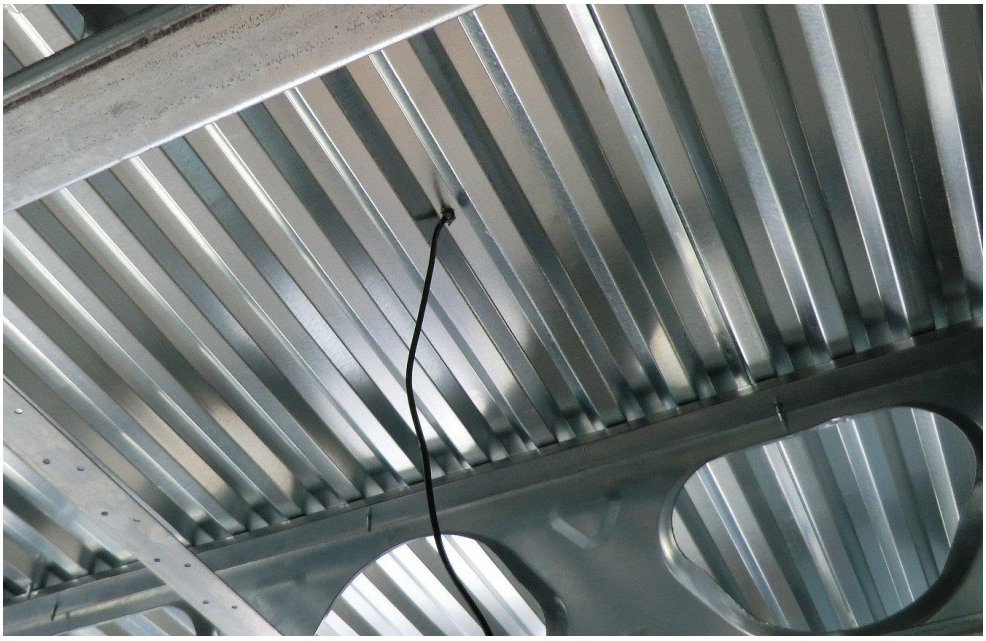
We made a construction drawing showing the location of each sensor on the floor plan. All of the sensors were labeled to insure that our sub put them in the correct locations.



The basement slab sensors were simply tied directly to the re-mesh with cable ties.



The main level slab sensors were installed by drilling a hole through the metal pan and dropping the cable down into the basement. A small dab of silicon caulk was used to prevent the cable from rubbing against the metal during the pour.



I used Cat-5 cable as the backbone to connect all the sensors together. Cat-5 cable consists of four (4) twisted pairs of 24 AWG solid copper wire (i.e. four pairs equals eight wires). This is overkill since the *1-Wire* channel only needs three wires but Cat-5 is inexpensive and readily available at big-box stores.

The *1-Wire* bus is low voltage (5 volts) and all of the connections are inside the house. I used Scotch-Lock UR gel-filled connectors to make the connections.

4 Arduino Uno Software

The software for this project is configured specifically for our sensor layout but provides a good example of how to collect values from a large array of temperature sensors. The software is available through our web site at www.neoterra.us or you can e-mail the author at gagner1@comcast.net.

4.1 Arduino Development

The Arduino Uno runs open-source software based on the C/C++ programming language. Although I've always refereed to a program as a *program*, in the Arduino world a program is referred to as a ***sketch***.

The Arduino can only hold one *sketch* at a time. The *sketch* is written/developed on a PC using the ***Arduino Environment*** and then loaded into the Arduino. The Development Environment is free and can be downloaded from the Arduino site at http://www.arduino.cc/en/Main/Software

The current version of the Arduino Environment is 1.6.3. Our project was written and tested over one year ago (before the house was built) using version 1.0.4.

4.2 Temperature Monitor Sketch

A high level flowchart of the Arduino temperature monitor software is shown on the next page.

Like standard 'C', each Arduino sketch has a program entry point. Unlike 'C' the entry point is called setup() rather than main() and its primary purpose is the initialize the controller hardware.

In this project Setup() performs the following steps:

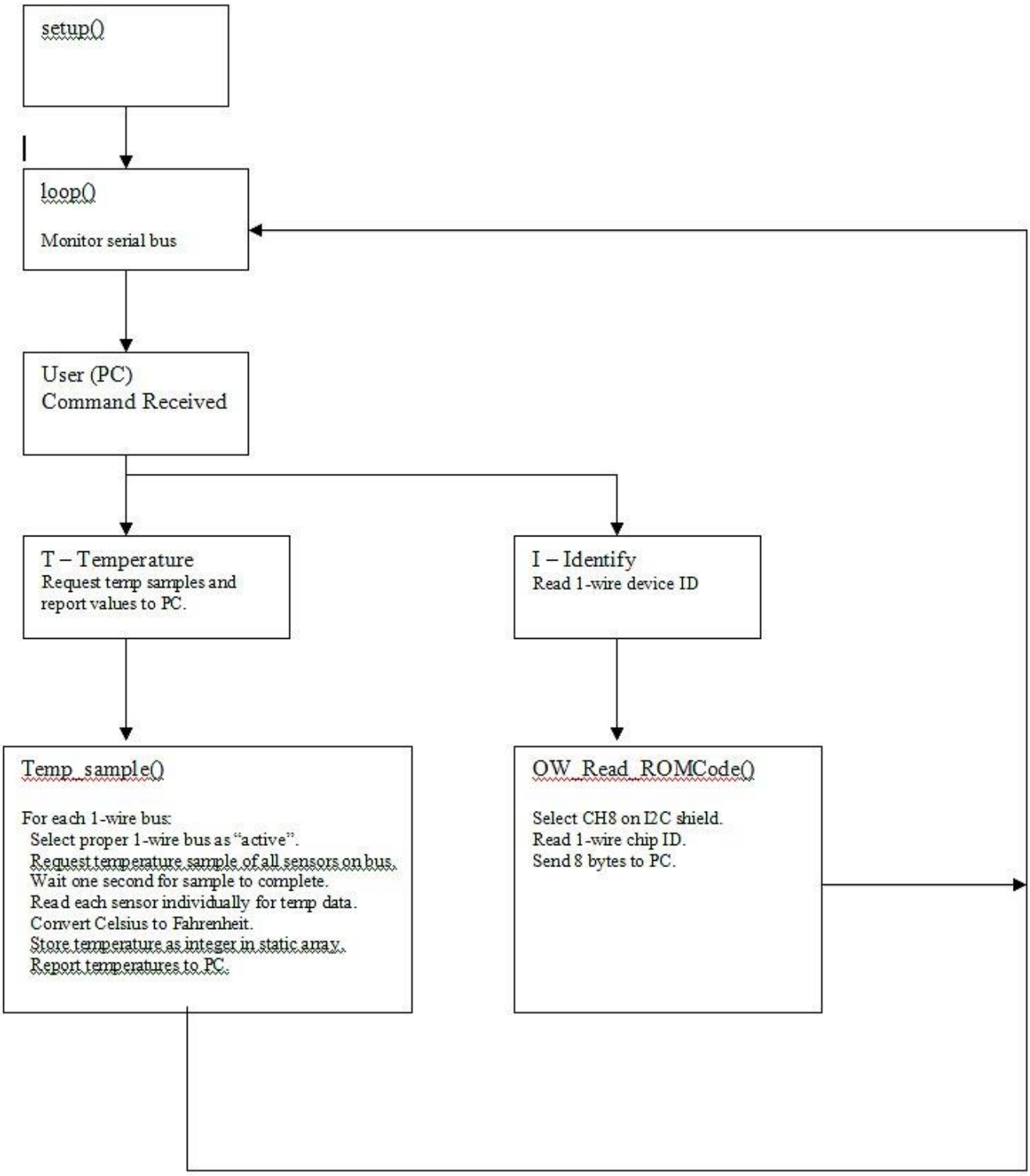
Set serial communication to the PC at 9600 baud
Configure as I2C Master. Analog pin 4 as SDA and Analog pin 5 as SCL
Reset the 1-Wire shield and configure for 1-Wire communication

Once setup() completes the program falls into the loop() function.

As the name implies, loop() will run forever. In this application the loop() function simply waits on an indication that a command has been received from the PC, determines if the command is valid and then executes the appropriate function.

Currently, there are only two valid commands:

- I=Identify. This is used to read the 64-bit address for a new 1-Wire device
- T=Temperature. This will read all of the attached sensors and return the values to the PC


setup()
loop()
Monitor serial bus
User (PC)
Command Received
T – Temperature
Request temp samples and
report values to PC.
I – Identify
Read 1-wire device ID
Temp_sample()
For each 1-wire bus:
Select proper 1-wire bus as "active".
Request temperature sample of all sensors on bus.
Wait one second for sample to complete.
Read each sensor individually for temp data.
Convert Celsius to Fahrenheit.
Store temperature as integer in static array.
Report temperatures to PC.
OW_Read_ROMCode()
Select CH8 on I2C shield.
Read 1-wire chip ID.
Send 8 bytes to PC.

4.2.1 Identify Command

The Identify (I) command is used to read the address of a new sensor before installing it.

The command will return eight (8) bytes to the PC, each byte on its own line, depicting the unique 64 bit ROM Code for a 1-Wire device. In order to be read, the 3 wires (DQ, VDD, and GND) of the 1-Wire device must be attached correctly to CH8 of the 1-Wire shield I/O bus. Only a single 1-Wire device may be connected at a time. Once read, the output on the PC monitor will look something like this:

```
ROM ID of 1-wire device is
28
79
F0
EB
 3
 0
 0
8F
```

Note values are reported as hexadecimal and that the least significant byte of the stream is reported first. These data may then be loaded into the source sketch as a static structure value to add the sensor to the project and communicate to the device individually. The data loaded into the structures has the least significant byte loaded into array position zero (0), which can have the appearance of being backwards but in fact is correct.

4.2.2 Temperature Command

The Temperature (T) command is the primary function of the project.

When the Arduino receives the T command it launches a function to request a fresh read of temperatures from all sensors and to report the temperatures back to the PC via the Monitor port.

The function is called Temp_sample(). The logic flow of Temp_sample is as follows:

- Enter a for loop that will step through each 1-wire bus in the project. In this case it will start with the Basement bus and step through to the Attic bus.
- A *Convert Temperature* command is broadcast to all 1-wire devices on the active bus. This command instructs all sensors to grab a fresh temperature reading and to store the read temperature value into their internal scratchpad (SP) memory. Since no value is actually read yet, this command is acted upon by all sensors simultaneously with no risk of 1-wire collisions.
- Since the sensors are set to 12 bit resolution (by default) the temperature conversion algorithm running in the sensor may take as long as 750 milliseconds – the higher the resolution, the longer the conversion. The code waits one (1) second for the conversion to complete.
- The code then accesses a static structure to determine the unique 64 bit ROM Codes for each device on the active bus and proceeds to query each sensor for temperature data.

- The sensors use two bytes in their SP memory to store temperature values. Byte 0 of the SP is the LSB of the temperature and byte 1 is the MSB. The temperature in the SP is stored as Celsius and if the value is negative, the most significant five bits of the MSB will be HIGH. The least significant four (4) bits of the least significant byte are fractions of a degree. These four bits are discarded. At this point, the code converts the temperature to Fahrenheit and stores the equivalent value as a signed int variable in a static array in host memory.

Once all of the 1-Wire channels have been processed the controller reads all of the temperature values from memory and converts the values to an ASCII string for communication to the PC.

Note that it takes the controller approximately 1 second to process each 1-Wire channel and there are currently four 1-Wire channels being used in this project. Thus, the PC must wait roughly four seconds to receive the results from the controller when it sends a T command.

A typical response would look something like this:

```
BS 64, 64, 63, 66, 65, 63, 64, 66
FS 69, 70, 69, 71, 70, 72, 69, 68, 68, 70, 71, 70
CS 76, 74, 76, 75
AS 58, 56, 48, 48, 54.
```

The response lists the temperatures of each sensor by channel where

- BS = Basement Sensors
- FS = First floor sensors
- CS = Ceiling sensors
- AS = Auxiliary sensors

5 Results

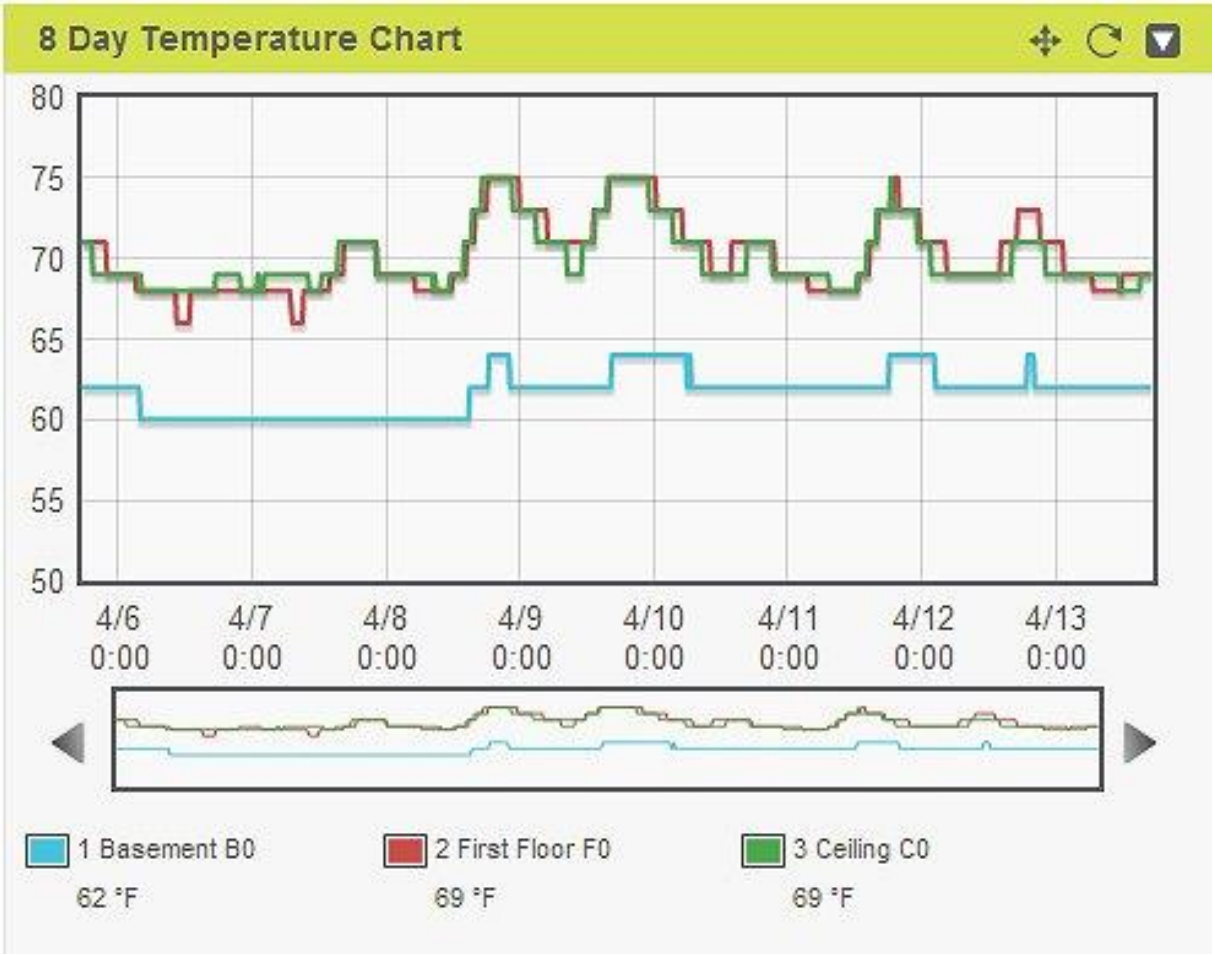
The sensors were installed in December of 2013 when the basement slab was poured. We brought the temperature monitoring system online in March of 2015 when we moved into the house. At this point, the sensors have been embedded in the concrete for 15 months with no ill effects.

We have been collecting data for several weeks. The following graph shows the temperatures for three sensors for the eight day period between April 6th-13th .

The sensors in this graph include:

- Green – Ceiling sensor in the Great Room
- Red – Floor slab sensor in the Great Room
- Blue – Basement slab sensor below the Great Room

Note that the basement is unheated during this period and the main level has a setpoint of 68 degrees. Since the space is heated the temperature will not fall much below 68 degrees.



The graph clearly shows the difference between cloudy days on the 6th and 7th versus sunny days on the 8th and 9th. On days with no sun the slab struggles to stay at 68 degrees (it is exposed to the unheated basement below). On sunny days the main level slab will reach 75 degrees.

It's interesting to note that the slab doesn't start to really warm up until late afternoon (4:00 – 6:00 PM). Following a sunny day the slab will stay above 70 degrees for up to 12 hours and the ceiling temperature stays above the 68 degree setpoint until late the following morning.

On April 20th we did an experiment where we set the heating temperature down to 58 degrees overnight. The 20th was mostly cloudy but the slab reached a temperature of 71 degrees by late afternoon. With no heat, the slab temperature dropped to 64 degrees over a 12 hour period between 8PM on the 20th and 8AM on the 21st. A bit more than ½ degree per hour.

Note that the ceiling temperature drops 1 – 2 hours before the slab temperature follows suit. This clearly shows how the slab is giving up heat into the space above.

6 Appendix A – Components and Vendors

Item	Qty	Price[1] (ea)	Description	Use	Vendor
1	1	$109.95	Intel DN2800MT Marshalltown Fanless Motherboard	Mini-ITX: PC interface to host micro-controller	Logicsupply.com
2	1	24.90	Arduino Uno	Host micro-controller	MC Pros
3	1	29.95	PCB: 1-wire Driver DS2482-800	I2C to 1-wire interface	InMojo.com
4	25	5.00	Waterproof stainless steel DS18B20.	Sensors embedded in concrete, placed in attic and outdoors.	YourDuino.com
5	1	18.03	3M: 3-wire UR Butt Splice connectors (100 pack)	Connections NOT made in concrete	Amazon.com
6	1	12.50	Proto Shield for Arduino Kit – v.5	PCB to mount I2C board to Arduino Uno	Adafruit.com
7	1	15.00	Clear enclosure for Arduino	Plastic enclosure box to mount Arduino and, shield, and RJ-45 jacks.	Adafruit.com

[1] May not include shipping or tax.