Remote Temperature Data Logging

Hunter Reid SUNY Cortland

PHY 495 Independent Study, first draft

July 30, 2020

Abstract:

Thermistors are a temperature dependent resistor that can be used to make temperature readings. When in series with a known resistance you can measure the voltage to find the resistance of the thermistor, and with some math, figure out the temperature of the medium you want to know. The Arduino's ability to read analog inputs and convert them to digital makes it a great data logging device. The board can be programmed to take readings from the thermistor and apply them to a formula to find the temperature. The purpose of setting up this Arduino with thermistors is to help with a more in depth experiment. The process of freezing ice in the ground when the air temp is above freezing with the use of night-sky cooling. The thermistors will be used to monitor the temperature throughout the day.

Introduction:

Thousands of years ago in Iran, they use to freeze water overnight using the process known as "night sky cooling" [9]. The Persians would fill throughs with water in the evening, and then awake the next morning before dawn to harvest the ice that froze over the night. This ice was then stored in "yakhchals" or ice huts, which are insulated pits that keep ice [9]. The fascinating part about this is the air temperature during the night would not dip below freezing but the water was able to reach freezing temperature [9].

Thermal radiation was is what allowed water to reach freezing temperatures, this is the process referred to as night sky cooling or radiative cooling [9]. Heat can flow from the surface of the earth into the depths of space through radiation due to outer space having a temperature of just a few Kelvin [8]. Deserts have low humidity and clear skies allowing for maximum heat transfer and little reflection back from our own atmosphere [9]. The amount of heat able to flow to space is greater than the heat reflecting back to earth, this drops the temperature of the water enough to freeze.

Night sky cooling can freeze water over night in the deserts of Iran, but is it able to in other climates? Can water be left over night in a climate such as upstate New York and freeze? Obviously in the winter temperature will dip bellowing freezing, but the question is if water can be frozen when air temperature is above freezing in upstate New York. New York has variable weather including humidity and cloud cover. Even if water cannot be frozen there must still be some effect of night sky cooling. Testing the effect will take some planning, and some tools, most importantly to monitor temperature. To measure and save temperature readings throughout the night or even over a span of days will take some

important components. A temperature sensor or thermistor is needed to take temperature readings, a microcontroller is needed to take the measurements, a solid-state memory drive is needed to store the data to, and a power source is needed to run the system.

Programming an Arduino to take temperature readings with a thermistor has reason besides just taking a temperature. If temperature is needed a thermometer could be used to simply measure the temperature. The benefit of using a microcontroller is it can be programmed to take multiple readings over a period of time, or have something happen when a temperature hits a certain point. It would be tedious and unrealistic to sit there and manually take periodic readings for days on end. This makes microcontrollers very useful tools to scientists and engineers. Thermistor stands for thermal resistor, which is a resistor that is sensitive to temperature [1]. This means that when temperature changes so does the resistance of the thermistor. There are two types of thermistors, NTC (negative temperature coefficient) thermistor and PTC (positive temperature coefficient) thermistor [2]. For a NTC thermistor resistance goes down as temperature rises, and for a PTC thermistor resistance goes up as temperature rises [3]. NTC thermistors are more widely used for temperature measurements than PTC thermistors [1]. PTC's are generally used as "resettable fuses" because as they get hot the resistance increases and "choke back the current" [1]. From here on out when talking about thermistors it will refer to NTC type. Thermistors are obviously not the only type of temperature sensors on the market, but they are very widely used. The main reason that thermistors get used often is their low cost coupled with accuracy; thermistors can measure with an accuracy of plus or minus .25 degrees Celsius [1]. They are also rugged instruments that can withstand extreme conditions and be protected with epoxy or glass coatings to last a long time [2]. There are different thermistors for different applications. The epoxy is usually used for lower temperatures (-50 to 150 degrees Celsius), while the glass is used for higher temperatures (-50 to 300 degrees Celsius) [2]. Not only are thermistors differed by coatings but by their actual resistance as well. Two of the most common used thermistors are 10k and 100k thermistors; these values are the thermistors resistance at 25 degrees Celsius [3]. Lower temperature applications usually use thermistors in the 10k region while higher temperatures use thermistors in the 100k region [2]. If you use a 100k thermistor it will use much less current than a 10k thermistor (V=IR), so if you need a battery to power a system for a long time you may want to use a 100k thermistor which will draw less current.

Theory:

Since a thermistor's resistance is temperature dependent all that is needed is the Steinhart-Hart equation (equation 6) to convert it to temperature once the resistance of the thermistor has been found. However, it is not quite that simple because microcontrollers, like the Arduino Uno, cannot measure resistance [4]. The circuit is a simple circuit containing a resistor of known resistance (usually 10k or 100k) in series with a thermistor (matching resistance at 25 degrees Celsius). The microcontroller can take a voltage reading between the resistor and thermistor, which will change with temperature because the thermistors resistance values changes with temperature. This is an analog reading that the Arduino will convert into a digital

reading. The digital reading can be used to find the resistance of the thermistor starting with the following equation.

$$V_{out} = \frac{R}{R + 10K} * V_{in}\ (equation 1)\ [4]$$

V_{out} is the voltage reading, V_{in} is the input voltage, R is the unknow thermistor resistance, and 10K is the resistance of the known resistor. The V_{out} is converted to a digital reading so to find the analog to digital conversion (ADC) the following equation is needed.

$$ADC = V_{out} * \frac{1023}{V_{aref}}\ (equation\ 2)\ [4]$$

ADC is the digital voltage reading, V_{out} is the analog voltage reading going into the microcontroller, 1023 is the 10 bit ADC conversion, and V_{aref} is the analog reference voltage which is the same as V_{in}. Once plugging V_{out} into the ADC equation.

$$ADC = \frac{R}{R + 10K} * V_{in} * \frac{1023}{V_{aref}}\ (equation\ 3)$$

$$ADC = \frac{R}{R + 10K} + 1023\ (equation\ 4)\ [4]$$

V_{in} and V_{aref} are the same so they cancel leaving a simple ADC equation that can be solved for R.

$$R = \frac{10k}{\frac{1023}{ADC} - 1}\ (equation\ 5)\ [4]$$

Now that the resistance can be found with an ADC value measured by the Arduino the resistance can be plugged into the Steinhart-Hart Equation.

$$\frac{1}{T} = \frac{1}{T_0} + \frac{ln\left(\frac{R}{R_0}\right)}{B}\ (equaiton\ 6)\ [4]$$

T is the temperature to solve for, T_0 is room temperature in Kelvin (298.15), B is the coefficient of the thermistor, R is the measured resistance, and R_0 is the resistance at room temp (i.e. 10k or 100k depending on your thermistor).

Methods:

1. **Night Sky Cooling:**

Being in upstate New York the goal is to try and freeze water using night sky cooling when air temperatures are above freezing. However, just getting some data on night sky cooling in this part of the world will suffice. The idea is to dig holes into the ground, exposing them to the sky at night and covering them from the sun during the day. Each hole will have different forms of insulation (wood, soil, foam), but the same volume and shape. The thermistors will be in the holes taking temperature readings throughout the course of the experiment. These temperature readings will allow us to monitor how much radiative cooling takes place. Upstate New York, unlike the desert, has much more variable weather including four seasons, cloud cover, and humidity. Carrying out the experiment during different stages of weather and different seasons of the year could yield quite different results.

The phenomenon of night sky cooling is being research for renewable energy applications. Currently 17% of global electricity use goes toward cooling systems, and that contributes to 8% of global greenhouse gas emission [10]. The problem with night sky cooling is that it is at night, and most cooling systems are needed during the day [10]. Research is being done on taking advantage of night sky cooling not only at night but during the day [10]. Not only is night sky cooling an amazing phenomenon allowing water to freeze with warmer air temperature, but it is leading to research to help fight climate change.

2. **The Code**

The purpose of this set-up is to lead to a bigger experiment using radiative cooling to freeze water in the ground at night. The thermistors will be in the experiment to take periodic temperature readings to show progress of cooling down the system. The thermistors are programmed to take a reading every 5 minutes and store the data to a SD card using the Arduino IDE. The Arduino takes an analog reading and stores an ADC value on the SD card, which, later on can be uploaded to an excel file and converted to temperature values. The Arduino is fully capable of doing the math to convert the ADC value to a temperature, but there just leaves more room for error. Since the experiment may take a few days of measurement having less chance of incorrect values is worth uploading to an excel file and converting there.

The codes main purpose is to take period analog reading, convert them to digital form and store the values on an SD card. These digital values can then be manipulated into temperature values. For more precision 10 analog readings will be taken and averaged at every time interval.

Since there will be periodic measurements an “if statement” function inside a loop function will be most useful. The Arduino’s internal clock is accessed through the

"millis()" function [6]. Once a desired time is hit the use of another loop function, in this case a "for" loop, is useful once again to make the ten readings. Making analog readings in Arduino requires the "analogRead" function [7]. Values of the readings can be saved to a variable, and the variable can then be written to the SD card. To save to the SD card, the card needs to be opened, printed to, and then closed. To open the SD card the "filename=SD.open()" function is used, to print to the SD card the "filename.print()" function is used, and to close the SD card the "filename.close()" function is used [5]. (CODE FOR THIS SET-UP IN LOCATED IN THE APPENDEX).

3. Circuit

Thermistor circuits are pretty simple containing the thermistor and known resistor in series with an analog pin in between them as shown in the schematic below (Figure 1.). For this particular set-up an Arduino Uno was used as the microcontroller, an Adafruit 10k thermistor for the thermistor, 10k series resistor, and a HiLetgo shield was used for the SD card. A wire was inserted into the 5V pin on the board leading to 4 10k resistors. After the resistors the circuit had a wire leading to an analog pin. Then the thermistors were placed in series with their respective loop and fed to ground.

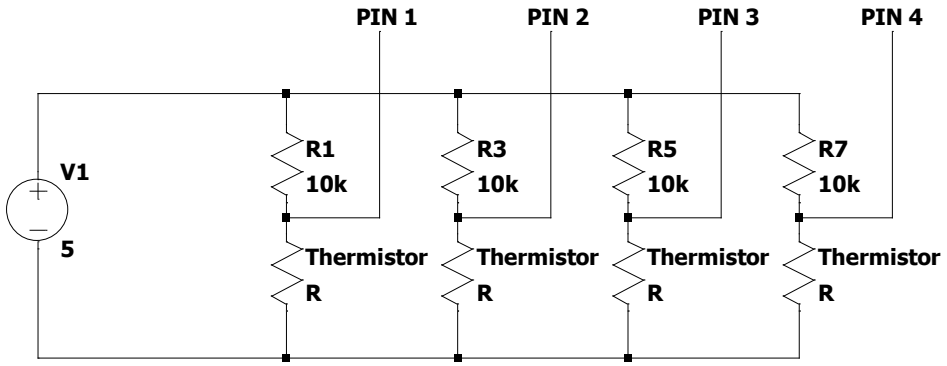

Figure 1. (circuit design)

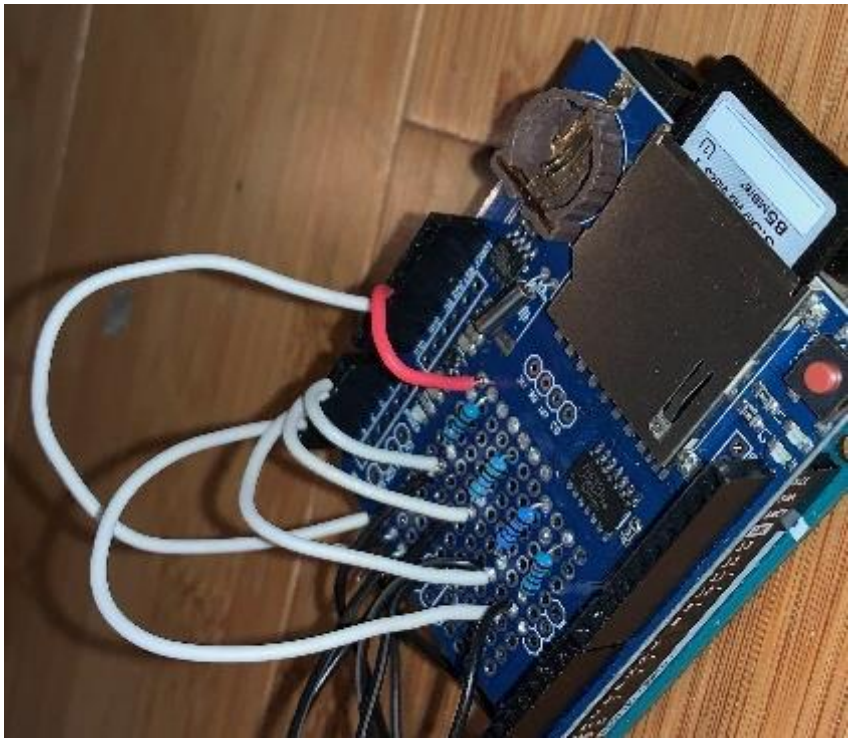

Figure 2. (Built circuit on HiLetgo shield

4. Debugging

Looking at the code there does not seem to be much to it, but as many know, problems always arise when programming. Rarely does one write a code and have it run as needed right away. Originally this code was written having and “if” and “else” statement for each thermistor. This made the code a bit clunky and hard to follow as well as difficult to integrate the clock. This led to switching to a “two loop” system were there was and “if” statement and “for” loop within the main loop. Once this structure was set problems arose with writing to the SD card. The issue boiled down to improper syntax due to unfamiliarity with the language. Too much time was spent trying to figure out why the program would not write to the SD card, which should be a trivial task, and the issue was that the file name being created was too many characters.

Along with coding difficulty there was hardware difficulty. Inconsistent thermistor readings kept arising. The first problem was simply a few soldering errors where that was not a good connection. Once that was solved three of the thermistors gave readings that were consistent with each other and accurate. The fourth kept getting readings that were not accurate. This ended up being a problem with the shield. The “A4”

analog pin would not transfer properly to the Arduino board. When the pins were directly plugged into the board the thermistors all read consistent accurate measurements. With the shield only the first three pins continue to have acceptable readings. This lead to some alterations of the board and shield; the first three pins got plugged into the shield while the fourth pin got put directly into the Arduino board.

5. Data

To make sure the thermistors are working properly a couple calibration tests were made. The first was done by making an ice bath (Figure 3.), this gives a Theoretical temperature of 0°C / 32°F. The second test, water was boiled on the stovetop giving a theoretical temperature of 100°C / 212°F (Figure 4.). The results can be seen in Figure 5.

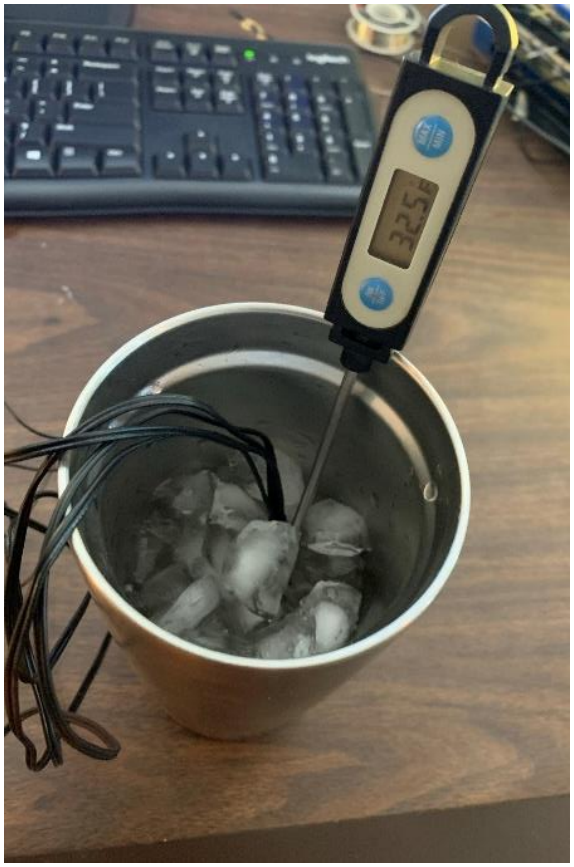

Figure 3. (ice bath calibration)

Figure 4. (boiling water calibration)

	DIGITAL READING	CELCIUS MEASURED	FAHRENHEIT MEASURED	FAHRENHEIT ACTUAL	PERCENT ERROR
ICE BATH	776.75	1.36	34.45	32.5	5.99%
BOILING WATER	69.33	99.15	210.50	210	0.22%

Figure 5. (Data from calibration tests)

Conclusion:

Having temperature dependent resistors makes for an easy cheap way to accurately monitor temperature levels with a microcontroller [1]. Knowing the task at hand to decide what type of thermistor to use is always important, but the great thing about thermistors is their simplicity. This allows them to be used with microcontrollers for many different temperature dependent applications. If there is going to be a lower temperature a 10k thermistor with epoxy coating rather than a 100k glass coated thermistor is the better route [2]. Also, knowing the power supply might depict which thermistor is the best choice based on how much power will be used. The 100k thermistor will pull lest current, thus making the battery last longer.

There are plenty of web pages that give the code to run a thermistor with an Arduino. However, it is beneficial to write the code from scratch so that every step is known in detail. This will allow for easy debugging, appending, and addition of new tasks. Not every code for thermistors and Arduinos will work for every setup. Styling and formatting the code to each setup can be crucial for the different tasks at hand. Arduinos website has pages with all the different available syntax and their applications. Making the code teaches the language and process for programming to help future work. Understanding the code is vital if something goes wrong and needs to be fixed.

For this particular code some modifications to increase its usability would be to integrate the RTC. The internal clock help print time stamps based on when the program started running. The RTC would show the exact time of day the reading was made. To integrate the RTC a battery is needed in the shield, and the code would need to be updated. RTC libraries would need to be accessed and the time and date would need to be set.

For this particular setup the thermistors were compared to a thermometer with ice water and boiling water. The thermometer read 32.5°F in the ice water, and the thermistors measured 34.4°F. Theoretically the temperature of and ice bath should be 32 degrees, but it would have been better to use more than one thermometer to compare the thermistors to. There are a few factors that could lead to the thermistors being off 2 degrees. One being the coating on the thermistor could have insulated it from the true temperature though I doubt this coating had much effect. 5 measurements were made spaced 5 minutes apart giving the thermistors plenty of time come to equilibrium temperature with the water. The second option is that the Steinhart-Hart equation used has some errors. The constants in the equation used could be approximated too much, or the coefficient for the thermistor is not accurate enough at measured the measured temperatures. NTC thermistors increase non-linearly, which is why the Steinhart-Hart equation

uses a logarithm [2]. Adafruit supplies a table with temperature/resistance measurements that are more accurate [11]. The third, and final, area of error is that the thermistor is not exactly 10k ohms at room temperature skewing the readings more noticeably as they creep further from 25°C. The thermistor coefficient of 3950, and room temperature resistance of 10k have a 1% fluctuation [11]. However, in boiling water the thermometer read 210°F, and the thermistors read 210.5°F. This appears to be a very acceptable result for temperature measurement.

Moving on from here will be the use of "night sky cooling" trying to freeze water in the ground when air temperature is above freezing. This this was done thousands of years ago by the Persians to make ice overnight in the deserts of Iran through thermal radiation between earth and space [8]. New York has variable weather and much more cloud cover; the process of freezing water will be much less likely, but observations of night sky cooling could still be observed.

References:

[1] Thermistor Overview. July 29, 2012. Adafruit; [July 29, 2012].

https://learn.adafruit.com/thermistor

[2] What is a Thermistor and how does it work?. Copyright 2003-2019. Omega Engineering Inc.

https://www.omega.com/en-us/resources/thermistor

[3] How Do Thermistors Work. Copyright 2020. Leaf Group Media.

https://sciencing.com/thermistors-work-4709009.html

[4] Using a Thermistor. July 29, 2012. Adafruit; [July 29, 2012].

https://learn.adafruit.com/thermistor/using-a-thermistor

[5] Using the SD library to read and write to a file on a SD card. 2020 Arduino. [2015/08/18].

https://www.arduino.cc/en/Tutorial/ReadWrite

[6] millis(). 2020 Arduino.

https://www.arduino.cc/reference/en/language/functions/time/millis/

[7] analogRead(). 2020 arduino. [2019/07/02].

https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

[8] The Process of Freezing Ice in the Desert According to Physicists. Copyright 2018. Tecped.com.

https://tecped.com/process-of-freezing-ice-in-the-desert-according-to-physicists/

[9] Persian ice house, or how to make ice in the desert. Copyright 2020. Field Study of the World.

https://www.fieldstudyoftheworld.com/persian-ice-house-how-make-ice-desert/

[10] How we can turn the cold of outer space into a renewable resource. April 2018. TED Conferences, LLC.

https://www.ted.com/talks/aaswath_raman_how_we_can_turn_the_cold_of_outer_space_into_a_renewable_resource#t-98957

[11] 10K Precision Epoxy Thermistor - 3950 NTC. Adafruit

https://www.adafruit.com/product/372

Appendix:

(explanation of code format is in methods → 2. The Code.)

```
#include <SPI.h> //library for SD

#include <SD.h> //library for SD

File myfile; //creating file for SD
int HumidityPin= A0; //assigning pin to humidity sensor
int ThermistorPin1=A1; //assinging pin to thermistor
int ThermistorPin2=A2; //assinging pin to thermistor
int ThermistorPin3=A3; //assinging pin to thermistor
int ThermistorPin4=A4; //assinging pin to thermistor
int r=10000; //resistor value
const int SDcard=10;
unsigned int samples=10; //amount of measurments per test
float V1; //pin 1 voltage measurement
float V2; //pin 2 voltage measurement
float V3; //pin 3 voltage measurement
float V4; //pin 4 voltage measurement
const unsigned long Time=300000; //time between tests
unsigned int long previousTime=0; //updated time for tests


void setup() {
  // put your setup code here, to run once:
```

```
  Serial.begin(9600);
  pinMode(SDcard,OUTPUT); //SD card pin
  pinMode(HumidityPin,INPUT);
  pinMode(ThermistorPin1,INPUT);
  pinMode(ThermistorPin2,INPUT);
  pinMode(ThermistorPin3,INPUT);
  pinMode(ThermistorPin4,INPUT);


  if (!SD.begin(SDcard)){//used to tell if the sd card in initiallized
    Serial.print("Could not initialized SD card.");
}
  else{
    Serial.print("SD card initialized");
    Serial.print(" ");

}
  myfile=SD.open("temp.txt",FILE_WRITE);
  myfile.print("testing 1,2,3");
  myfile.println();
  myfile.close();
}

void loop() {
  // put your main code here, to run repeatedly:
  unsigned long currentTime=millis(); //starting internal clock

  if(currentTime-previousTime >= Time){ //if statement to take measurements every 5 minutes
```

```
previousTime=currentTime; //updating previous time
unsigned Clock=previousTime/60000; //converting time from milliseconds to minutes
myfile=SD.open("temp.txt",FILE_WRITE);
myfile.print("Time= ");
myfile.print(Clock);
myfile.println();
myfile.print("V1     V2     V3     V4");
myfile.println();
myfile.close();
Serial.print("5");
for(int i=0; i<samples;i++){ //for statement to take 10 measurements

  V1=analogRead(ThermistorPin1);//readaing pin and writing to array
  myfile=SD.open("temp.txt",FILE_WRITE);
  myfile.print(V1);//printing reading to sd card
  myfile.print(" ");
  myfile.close();

  V2=analogRead(ThermistorPin2);
  myfile=SD.open("temp.txt",FILE_WRITE);
  myfile.print(V2);//printing reading to sd card
  myfile.print(" ");
  myfile.close();

  V3=analogRead(ThermistorPin3);
  myfile=SD.open("temp.txt",FILE_WRITE);
  myfile.print(V3);//printing reading to sd card
  myfile.print(" ");
```

```
    myfile.close();

    V4=analogRead(ThermistorPin4);
    myfile=SD.open("temp.txt",FILE_WRITE);
    myfile.print(V4);//printing reading to sd card
    myfile.println();
    myfile.close();

    delay(10);

  }
 myfile.close();//closing sd card

}



}
```