INSTRUMENTED COVID-19 MASK WITH TEMPERATURE SENSORS AND ALARM LED:

FARMINGTON, NEW MEXICO AREA OF THE NAVAJO NATION

Lilliona Benally

9th Grade

Navajo Preparatory School

1220 W Apache St, Farmington NM 87401

PO Box 2147 , Fruitland NM 87416

lilliona.benally@navajoprep.com

Advisor: Yolanda Flores

606 ½ N. Dustin Ave.

Farmington, NM 87401

Signature:

RESEARCH PAPER

INSTRUMENTED COVID-19 MASK WITH TEMPERATURE SENSORS AND ALARM LED

Ms. Lilliona Benally

Navajo Preparatory School, Farmington, NM

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INSTRUMENTED COVID-19 MASK WITH TEMPERATURE SENSORS AND ALARM LED

Ms. Lilliona Benally
Navajo Preparatory School, Farmington, NM
Teacher: Ms. Yolanda Flores
Mentor: Dr. Daniel Winarski, IBM Retiree, Tucson, AZ

ABSTRACT

Because of the devastating effect that the COVID-19 pandemic had on the Navajo Nation and across the world, I decided to prototype and instrumented a respiratory mask having six temperature sensors in order to detect the temperature of the person wearing it and the health condition. This was mainly inspired off of the condition of the Navajo Nation right now as in some areas and some places some are not able to get tested or know their own health condition due to distance or lack of health care. These six temperature sensors were employed in three pairs; two inhale temperature sensors, two exhale temperature sensors, and two skin temperature sensors. These temperatures were monitored by an Arduino UNO microprocessor used with the Arduino Uno program and displayed on the serial monitor of my laptop with the USB cable. If the wearer's skin temperature exceeded a preselected value, indicating a fever, a red alarm LED lit, indicating the need to seek medical attention or further inspection. The mask gives needed protection to the wearer as mandated by law to wear mask when outside. However,

with this experiment, I was able to build and instrumented a respiratory mask with temperature sensors that can monitor and detect temperature changes and health of the wearer using an analog system and how I could look at it at different spans of time easily.

INTRODUCTION

Regretfully, the use of protective air-filtration masks was highly politicized and many people refused to wear them. Even when people did wear masks, these masks were passive, indicating that although the masks did help reduce the airborne transmission of Covid-19, the masks themselves gave no indication of the health of the wearer. My project was to prototype a respiratory mask with temperature sensors, to measure the temperature of the inhaled air, the temperature of the exhaled air, and the skin temperature of the wearer of the mask. The goal was to give the user an indication of his or her health, while offering protection against Covid-19.

PROBLEM

The problem was to identify how to protect the health of people and measure their health at the same time.

HYPOTHESIS

Can a respiratory mask be instrumented with temperature sensors and monitored by an Arduino UNO microprocessor, to give an indication of the health of the wearer?

VARIABLES

The variables in my experiment are:

- Inhaled air temperature measured at the intake filters: T0 and T1.
- Exhaled air temperature measured at the exhaust port: T2 and T3.
- Skin temperature measured at or near the chin of the wearer: T4 and T5.
- Sample period of the Arduino UNO, in seconds.

BACKGROUND RESEARCH

The COVID-19 pandemic has had a tremendous impact on the Navajo Nation, in the four-corners region of the desert southwest, to include total weekend lockdowns from Friday, 8pm, to Monday, 5am. The Navajo had 2,304.41 cases of Covid-19 per 100,000 people at its peak in May, compared to the New York state rate of 1,806 cases per 100,000, according to data from Johns Hopkins University [ref.1]. Since the pandemic began up until November 22, the Navajo Nation has had 15,039 cases out of its population of 173,667 from the 2010 census, which means this hard-hit population has over 8,659 cases per 100,000 people.

In the autumn of 2020, Navajo Preparatory School has been totally remote. Since the Navajo Nation is mostly rural with many people living in traditional Hogans, students of Navajo Preparatory School are definitely on the other side of the “digital divide." The connectivity we take for granted is minimal to nonexistent within the Navajo Nation, causing a disruption of our educational effort.

Symptoms of covid-19 include fever or chills, cough, shortness of breath or difficulty breathing, fatigue, muscle or body aches, headache, new loss of taste or smell, sore throat, congestion or runny nose, nausea or vomiting, or diarrhea [ref.2].

It is my goal to design a prototype respiratory mask instrumented with temperature sensors, to give an indication of the health of the wearer, while the mask itself helps to protect the wearer.

MATERIALS AND EQUIPMENT

The following are the materials and equipment and experimental procedures used.

- 3M Respiratory Mask
- Arduino UNO microcomputer
- Arduino UNO protective clear plastic case
- Six TMP 36 temperature sensors [ref.3]
- Red (positive), Black (negative, ground), and Yellow (signal) jumper wires
- Red Light Emitting Diode (LED)
- 330 ohm protection resistor for the LED
- USB cable
- Laptop
- Arduino IDE version 1.8.13 [ref.4]
- Soldering iron and solder
- Insulating electrical tape

EXPERIMENTAL PROCEDURES

A. Download the Arduino IDE (Integrated Development Environment) [ref.4]. This is available for Windows, Mac OS X, and Linux operating systems. The environment is written in java and includes other open source software.

B. Connect the Arduino UNO to the temperature sensors, alarm LED, and laptop as shown in Figure 1, below. The six temperature sensors shown in Figure 2 have a single power supply wire (red) and a single ground wire (black). The alarm LED uses a 330 ohm protection resistor to limit the electrical current through the delicate light emitting diode.

C. Bring up the Arduino IDE by clicking on the Arduino icon (green circle with an infinity sign, with + and – symbols). Bring up "tools" from the toolbar to interface the laptop and the Arduino UNO. As shown in Table 1, pick a board (Arduino UNO) and a port (assuming you have an Apple iMAC). Also, pick the Serial Monitor for display of the measured temperatures.

D. The six TMP temperature sensors are mounted in a 3M Respirator Mask, Figure 3, as shown in Figure 4. These temperature sensors are mounted in three pairs. Temperature sensors T0 (left filter) and T1 (right filter) measure the temperature of the inhaled air. Temperature sensors T2 and T3 measure the temperature of the exhaled air, at the exhale port of the 3M respirator. Temperature sensors T4 and T5 measure the skin temperature of the wearer near the chin area of the face.

E. Data is taken at a user selectable time interval, called a sample period. Right now, my Arduino Sketch is set to take data ever second, for demonstration purposes, and that data is shown in Figure 5. This sample period can be made longer if required.

F. In Figure 5, the inhaled air temperatures T0 (left filter) and T1 (right filter) were averaged to create inhale C. The exhaled air temperatures T2 and T3 were averaged to create exhale C. The difference between the hotter exhaled air and cooler inhaled air was calculated to be exhale C – inhale C = delta exhale-inhale C.

G. The skin temperatures T4 and T5 were averaged to create skin C. It was this average skin temperature which was used to trigger the lighting of the red alarm LED via the conditional statement "if (degreesC_skin > 38.) digitalWrite(LED_alarm, HIGH)." This conditional statement meant that if the skin temperature indicated a fever, then the LED_alarm pin (which was set to pin 13) would be send to high, meaning that power was applied to this pin to power the red alarm LED.

DATA, DIAGRAMS, AND OBSERVATIONS

The following figures are illustrative of the experimental procedure that was followed.

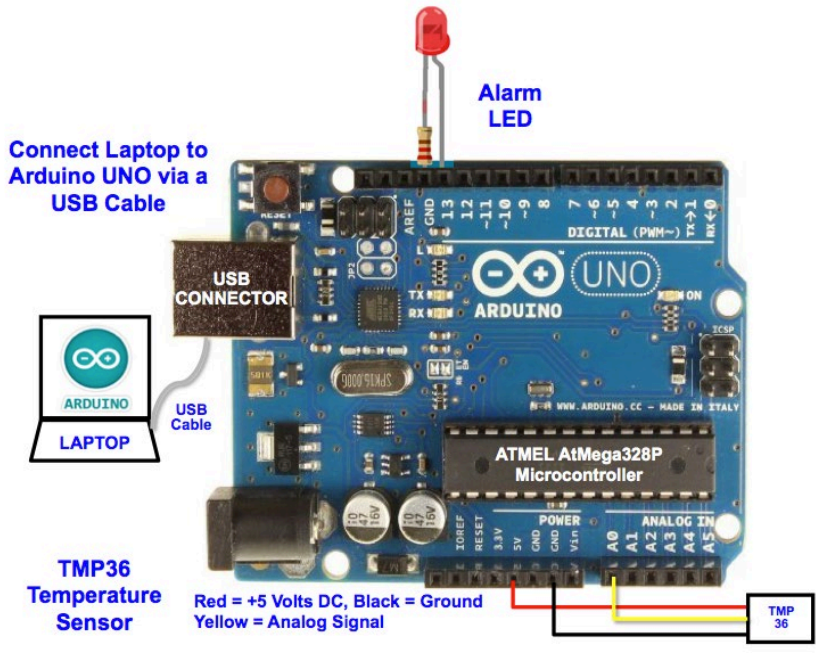


Figure 1. Arduino UNO connected to Laptop, Temperature Sensors, and Alarm LED

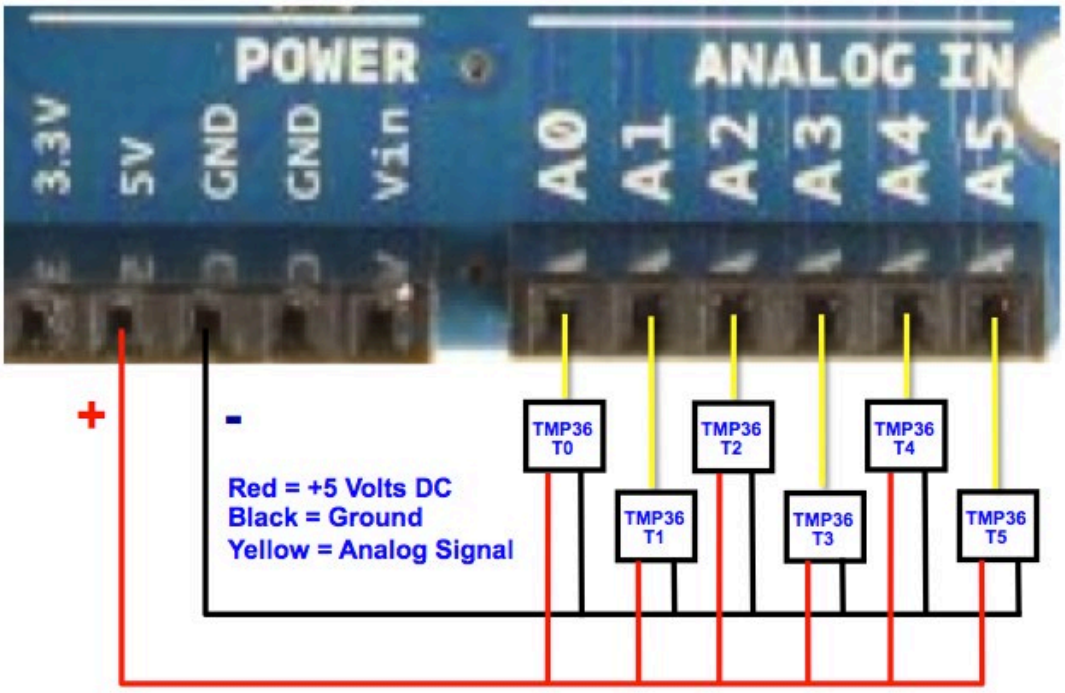
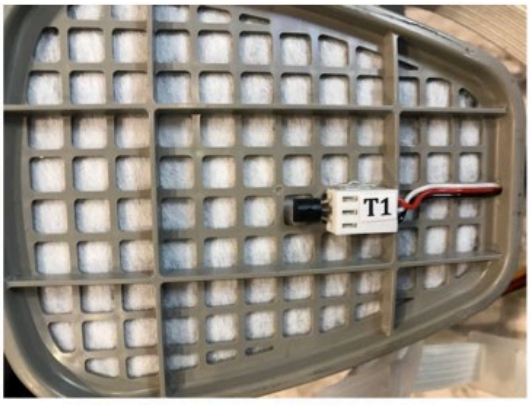


Figure 2. Six TMP36 Temperature Sensor Connections

Table 1. Interface Settings for the Arduino UNO and Laptop

TOOLS	SETTING	COMMENTS
Auto Format		
Archive Sketch		
Fix Encoding & Reload		
Serial Monitor		Select for Display of Temperatures
Board >	Arduino UNO	
Serial Port >	/dev/cu.usbmodemFD121	USB communication between iMAC and Arduino UNO

Figure 3. 3M Respirator



Right Filter Inhale Sensor T1
(Left Filter Inhale Sensor T0)

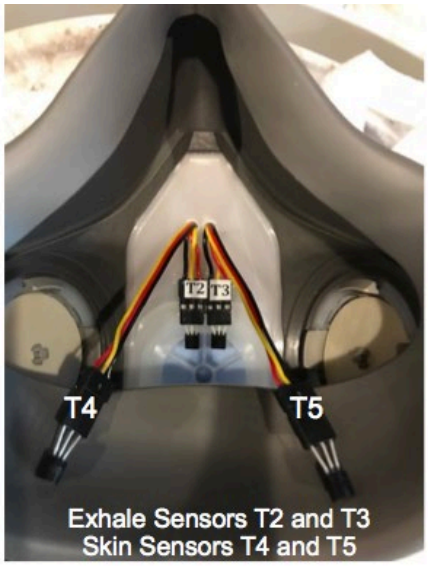


Figure 4. Locations of Six Temperature Sensors

DATA

The data taken is shown in Figure 5, below. The exhale air temperature was up to 7.08ºC (12.7ºF) hotter than the inhale air temperature. Also, the skin temperature was 33.98ºC (93.2ºF), indicating a healthy wearer. This data indicates that my prototype instrumented Covid-19 mask was successful.

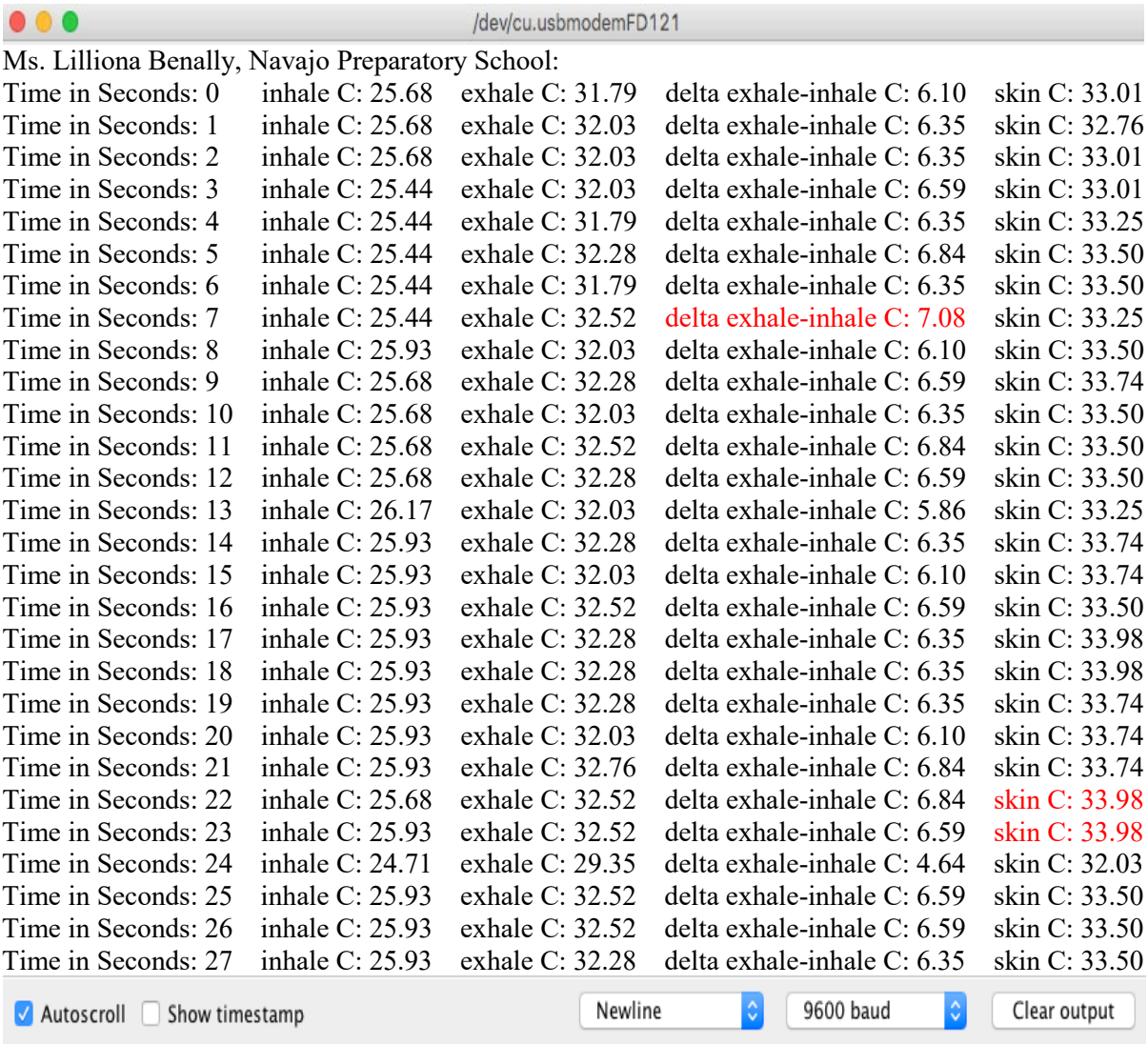


Figure 5. Data taken

Additionally, the red alarm LED would light up whenever the average skin temperature exceeded my threshold. In the Arduino sketch (Appendix A), I selected 38ºC as my threshold for a fever. However, this threshold can be reprogrammed for 28ºC for demonstration purposes, and by holding thermal sensors T4 and T5 in my warm hand, the red alarm LED will turn on.

CONCLUSIONS

My project showed that a Covid-19 mask could be instrumented, thus converting what would normally be a passive mask into an information gathering one. Thus, my hypothesis was accepted.

FUTURE DIRECTIONS

In the future, I could attempt to invoke Bluetooth Communications to relay data, rather than using an USB cable. I could also look for pressure sensors, to measure coughing, another indication of COVID-19.

REFERENCES

[1] "Navajo Nation faces devastating loss from Covid-19 pandemic," By Megan Marples, CNN Updated 3:22 AM ET, Tue November 24, 2020, https://www.cnn.com/2020/11/24/health/navajo-nation-coronavirus-losses-wellness/index.html

[2] Symptoms of Coronavirus, updated May 13, 2020. https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html

[3] Analog Devices: Low Voltage Temperature Sensors – TMP35/TMP36/TMP37. http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/TMP35_36_37.pdf

[4] Download the Arduino IDE (Integrated Development Environment) version 1.8.13. November 20, 2020. https://www.arduino.cc/en/Main/Software

[5] Atmel Atmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash Datasheet https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[6] Arduino Project Book, May, 2013. https://www.arduino.cc/starterkit

ACKNOWLEDGEMENTS

This project could not have been possible without the help of my teacher, Ms. Yolanda Flores who helped me throughout my project to help me understand computer programming. She pushed me to do the best I can. She was a very big support in the science aspect of school.

I would like to also thank Dr. Daniel Winarski, and Tyson Winarski, who mentored me throughout this project. They were more than willing to answer questions and offer advice throughout this project.

Throughout this project, with the help of these three individuals, I learned so much about programming an Arduino UNO, using the Arduino UNO to gather experimental data, and how to go about a formal science experiment. But I would also like to acknowledge my family who motivated me to stick with this project when I was stuck at a roadblock. They were supportive, driving me to the local libraries, etc. Thank you to everyone else who contributed in his or her own way.

Appendix A: Arduino UNO Sketch (Program)

This first page of my Arduino UNO Sketch has text bracketed by a leading /* and then a trailing */, indicating that all the information bracketed between consists of explanatory comments and not code that is compiled or executed.

```
/*

Ms. Lilliona Benally
Navajo Preparatory School
February 12, 2021

Go to TOOLS and make these selections AFTER connecting the Arduino UNO to the
laptop via USB cable:
1) Select BOARD (such as Arduino UNO)
2) Select PORT  (to match Arduino UNO)
3) Select SERIAL MONITOR to display temperatures on LAPTOP.  Set for 9600
BAUD.

SIX TEMPERATURE SENSORS:  T0, T1, T2, T3, T4, and T5.
  More information on the sensor is available in the datasheet:
  http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/TMP35_36_37.pdf

  When looking at the flat side of the temperature sensor
  with the pins down, from left to right the pins are: 5V, SIGNAL, and GND.

  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 0 for T0 temperature sensor
  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 1 for T1 temperature sensor
  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 2 for T2 temperature sensor
  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 3 for T3 temperature sensor
  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 4 for T4 temperature sensor
  Yellow Wire:  Connect the SIGNAL pin to ANALOG pin 5 for T5 temperature sensor

  Common Red Wire:   Connect to 5 Volts (5V) on the Arduino for all TMP36 sensors
  Common Black Wire: Connect to ground (GND) for all TMP36 sensors

*/
```

```
  const int temperaturePin0 = 0;  // analog pin 0 assignment
  const int temperaturePin1 = 1;  // analog pin 1 assignment
  const int temperaturePin2 = 2;  // analog pin 2 assignment
  const int temperaturePin3 = 3;  // analog pin 3 assignment
  const int temperaturePin4 = 4;  // analog pin 4 assignment
  const int temperaturePin5 = 5;  // analog pin 5 assignment

  const int LED_alarm = 13;        //  LED_alarm assigned to pin 13
  int time = 0;       // variable to store time (in seconds)
  int seconds = 1;   // incremental jump in time, in seconds.  300 seconds = 5 minutes.

void setup()
{
  Serial.begin(9600);
  pinMode(LED_alarm, OUTPUT);  //initialize digital pin 13 LED_alarm as an output.
  Serial.println(" Ms. Lilliona Benally, Navajo Preparatory School: ");
}
void loop()
{

  int sensorval0 = analogRead(temperaturePin0);
  int sensorval1 = analogRead(temperaturePin1);
  int sensorval2 = analogRead(temperaturePin2);
  int sensorval3 = analogRead(temperaturePin3);
  int sensorval4 = analogRead(temperaturePin4);
  int sensorval5 = analogRead(temperaturePin5);

  float voltage0 = (sensorval0/1024.0)*5.0;
  float voltage1 = (sensorval1/1024.0)*5.0;
  float voltage2 = (sensorval2/1024.0)*5.0;
  float voltage3 = (sensorval3/1024.0)*5.0;
  float voltage4 = (sensorval4/1024.0)*5.0;
  float voltage5 = (sensorval5/1024.0)*5.0;

  float degreesC0 = (voltage0 - 0.5) * 100.0;
  float degreesC1 = (voltage1 - 0.5) * 100.0;
  float degreesC2 = (voltage2 - 0.5) * 100.0;
  float degreesC3 = (voltage3 - 0.5) * 100.0;
  float degreesC4 = (voltage4 - 0.5) * 100.0;
  float degreesC5 = (voltage5 - 0.5) * 100.0;

  float degreesC_inhale = (degreesC0 + degreesC1)/2;
  float degreesC_exhale = (degreesC2 + degreesC3)/2;
  float degreesC_skin   = (degreesC4 + degreesC5)/2;
```

```
  if (degreesC_skin > 38. ) digitalWrite(LED_alarm, HIGH); //turn LED alarm “on” if
fever > 38C

  Serial.print("Time in Seconds: ");
  Serial.print(time);
  Serial.print(" inhale C: ");
  Serial.print(degreesC_inhale);
  Serial.print(" exhale C: ");
  Serial.print(degreesC_exhale);
  Serial.print(" skin C: ");
  Serial.println(degreesC_skin);

  time = time + seconds;
  delay(1000*seconds); // Programmable time delay, in milliseconds.

}
float getVoltage(int pin)
{
return (analogRead(pin) * 0.004882814);
}
```

```
Sketch uses 4632 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 340 bytes (16%) of dynamic memory, leaving 1708 bytes for local variables. Maximum is 2048 bytes.
```

The statements in green, above, indicate that this Arduino sketch compiled successfully and that the sketch used 4632 bytes (14%) of the total of 32,256 bytes (31.5 kB) available in the Atmel Atmega328P 8-bit AVR Microcontroller [ref.5]. Also, the global variables used 340 bytes (16%) of the dynamic memory, leaving 1,708 bytes for local variables (maximum is 2048 bytes or 2 kB). In these calculations, it was assumed that 1 kB is 1024 bytes.

This code may be copy-pasted into an Arduino sketch by another user. However, one possible problem is that the Arduino compiler may not like “quotation” marks from the above Times New Roman font. The solution is to individually delete each quotation mark in each serial.print statement and re-enter it within the Arduino sketch.

Returning to the contents of my Arduino UNO Sketch, the following int(eger) "analog in" pin assignments, from 0 to 5, match the analog pins shown in Figures 1 and 2.

```
const int temperaturePin0 = 0;  // analog pin 0 assignment
const int temperaturePin1 = 1;  // analog pin 1 assignment
const int temperaturePin2 = 2;  // analog pin 2 assignment
const int temperaturePin3 = 3;  // analog pin 3 assignment
const int temperaturePin4 = 4;  // analog pin 4 assignment
const int temperaturePin5 = 5;  // analog pin 5 assignment
```

The LED_alarm is assigned to digital pin 13, as shown in Figure 1.

```
const int LED_alarm = 13;       //  LED_alarm assigned to pin 13
```

The variable "time" is initialized to zero, and the increment in time (the sample period) is set to one second, which can be changed by the programmer.

```
int time = 0;       // variable to store time (in seconds)
int seconds = 1;   // incremental jump in time, in seconds.  300 seconds = 5 minutes.
```

I now set the serial baud to 9600, and initialize digital pin 13 LED_alarm as an output.

```
void setup()
{
 Serial.begin(9600);
 pinMode(LED_alarm, OUTPUT);  //initialize digital pin 13 LED_alarm as an output.
 Serial.println(" Ms. Lilliona Benally, Navajo Preparatory School: ");
}
```

At this point, I enter an infinite loop, where temperature data is taken once each sample period. The first thing I do is to perform analogReads of pins 0-5.

```
void loop()
{

 int sensorval0 = analogRead(temperaturePin0);
 int sensorval1 = analogRead(temperaturePin1);
 int sensorval2 = analogRead(temperaturePin2);
```

```
int sensorval3 = analogRead(temperaturePin3);
int sensorval4 = analogRead(temperaturePin4);
int sensorval5 = analogRead(temperaturePin5);
```

I then convert the analog data from pins 0-5 into degrees Celsius.

```
float voltage0 = (sensorval0/1024.0)*5.0;
float voltage1 = (sensorval1/1024.0)*5.0;
float voltage2 = (sensorval2/1024.0)*5.0;
float voltage3 = (sensorval3/1024.0)*5.0;
float voltage4 = (sensorval4/1024.0)*5.0;
float voltage5 = (sensorval5/1024.0)*5.0;

float degreesC0 = (voltage0 - 0.5) * 100.0;
float degreesC1 = (voltage1 - 0.5) * 100.0;
float degreesC2 = (voltage2 - 0.5) * 100.0;
float degreesC3 = (voltage3 - 0.5) * 100.0;
float degreesC4 = (voltage4 - 0.5) * 100.0;
float degreesC5 = (voltage5 - 0.5) * 100.0;
```

I then average the temperatures from analog pins 0 and 1 to provide an average inhale temperature. Similarly, I average the temperatures from analog pins 2 and 3 to provide an average exhale temperature. Finally, I average the temperatures from analog pins 4 and 5 to provide an average skin temperature.

```
float degreesC_inhale = (degreesC0 + degreesC1)/2;
float degreesC_exhale = (degreesC2 + degreesC3)/2;
float degreesC_skin   = (degreesC4 + degreesC5)/2;
```

Since one indication of Covid-19 is a fever, I use this conditional statement to turn on the LED shown in Figure 1 if the skin temperature exceeds 38ºC. The threshold value of 38ºC can be adjusted by the programmer.

```
if (degreesC_skin > 38. ) digitalWrite(LED_alarm, HIGH); //turn LED alarm "on" if fever > 38C
```

I now print out the following data, as shown in Figure 5.

```
Serial.print("Time in Seconds: ");
Serial.print(time);
Serial.print(" inhale C: ");
Serial.print(degreesC_inhale);
Serial.print(" exhale C: ");
Serial.print(degreesC_exhale);
Serial.print(" skin C: ");
Serial.println(degreesC_skin);
```

Finally, I increment time by the sample period "seconds." Then my program delays for 1000*seconds before gathering the next set of thermal data. The delay is based on milli-seconds, hence I multiply the sample period "seconds" by 1000 to calculate that delay.

```
time = time + seconds;
delay(1000*seconds); // Programmable time delay, in milliseconds.
```

The Arduino Project Book [ref.6] was very helpful in programming the above code.