[[1]](#footnote-1)

Thermostat & Temperature Alternation Photoresistor Brightness Calibration

Author: Keaton Raymond

Team Member: Amyleila Mejia

*Abstract*—The purpose of this lab was to design a program that would read temperature, a program that would read light value, and a program that combines reading the two. The integration of a thermostat was used in the first program to indicate the surrounding temperature in degrees Celsius, display the three most recent temperatures and if the temperature was rising, falling or stable. The integration of a photoresistor was used in the second program to calibrate the brightness to a certain area and display the current brightness as a percentage. The third program’s goal was to alternate between the two programs, running each program for ten seconds.

# INTRODUCTION

T

HE first task that needed to be accomplished was displaying the three most recent temperatures in Celsius and whether the average temperature was rising falling, or stable, waiting 1.25 seconds between each temperature reading. The temperature readings were displayed on the top row of the LCD with the most recent reading on the far left, the temperature before that in the middle, and the oldest temperature on the far right. On the bottom row, if the average was 0.15 degrees or greater less than the previous average it displayed “falling” on the left side, if the temperature was 0.15 degrees or greater more than the previous average it displayed “rising” on the right side, and if the temperature was within 0.15 degrees of the previous average, it displayed “stable” in the middle.

The second task that needed to be accomplished required more user interaction. The user had to calibrate a photoresistor to the room they were in, pressing a button to record the dark light value then pressing the button again to record the light value of the brightest area in the room. After calibrating the photoresistor, the program would print the current brightness as a percentage between the two values recorded during calibration every 0.75 seconds. The calibration process would only be running one time.

The third and final task required there to be one program that combined both programs from the previous two tasks. The program would run the code for calibrating the photoresistor one time in the beginning then infinitely alternate between the code for displaying the temperature and if the average is rising, falling, or stable for ten seconds. It will then run the code for displaying the light value as a percentage for ten seconds.

## Rising/falling temperature sensor

To accomplish the first problem of displaying the temperatures, four float variables were used; tempRead (the newest temp), temp1 (the temp on the left), temp2 (the temp in the middle) and temp3(the temp on the right). The program would wait 1.25 seconds before reading the voltage from the temperature sensor and converting it to degrees Celsius using the following formula:

voltage = analogRead(A0) \* 4.8828125;

tempRead = (voltage - 500)/10;

That new temperature reading was then assigned to tempRead, then temp2 would become the new temp3, temp1 would become the new temp2 and the newest reading tempRead would become temp1. It is very important that the values are assigned in that specific order as to not leave any variables empty. The program would then print each temp 1-3 in their respective positions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | . | 1 | 5 |  | 2 | 1 | . | 2 | 5 |  | 2 | 2 | . | 0 |
| temp1 | | | | |  | temp2 | | | | |  | temp3 | | | |

*Figure 1:* LCD display shows the temperatures in proper sequence. Note the included identifier of which temperature is which variable displayed beneath the temperature was not included in the actual display. However, it was included in the figure for demonstration purposes.

The second problem of displaying whether the temperature was rising, falling, or stable required the program to calculate the average of the three temperatures and compare it to the average of the previous three temperatures. To do this the average of temp1, temp2, and temp3 was calculated and assigned to the value of the float variable newAvg. Using a series of if else statements, the program compared newAvg to another float variable oldAvg to see if the average was rising, falling or stable and printed the corresponding word on the bottom row. After printing the word, the value of oldAvg was assigned the value of newAvg because then for the next temperature read it would recalculate newAvg and compare that new average to the previous average, oldAvg.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | . | 1 | 5 |  | 2 | 1 | . | 2 | 5 |  | 2 | 2 | . | 0 |
|  |  |  |  |  | s | t | a | b | l | e |  |  |  |  |  |

*Figure 2:* Example of if the average of these three temperatures was within 0.15 degrees of the previous average.

## Light calibration system

To accomplish the first problem of calibrating the photoresistor, the user was prompted to press the button in the dark and when the button was pressed it recorded the value of the photoresistor in that instant to the variable assigned to the low position in the mapping function used to convert the raw value recorded to a percentage. That same process was used to record the bright value. Both assignments were done using the following do while loop but with different variables for each:

do

{

fromHigh = analogRead(A1);

} while(digitalRead(button) == HIGH);

Once the calibration was complete and the two values were recorded, the program entered an infinite loop of printing the brightness as a percentage every 0.75 seconds.

To accomplish the second problem of converting the brightness to a percentage and printing it, the map and constrain functions were used in an infinite loop. In this infinite loop, the program would read the brightness every 0.75 seconds, then the map function took whatever value that was read between the dark and bright values recorded during calibration and converted them to a value between 0 and 100 percent. The constrain function would ensure that the values would stay between 0 and 100 in case it read a value greater than the brightness value recorded during calibration.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | h | e |  | b | r | i | g | h | t | n | e | s | s |  |  |
| i | s |  | a | t |  | 7 | 4 |  | % |  |  |  |  |  |  |

*Figure 3:* This is an example of what the LCD display would say for the brightness being at 74% between the recorded bright and dark values.

## Alternating between temperature sensor and light value

To solve the first problem of only calibrating the light sensor once, after calibrating the light sensor the program enters an infinite loop of repeating the two codes for ten seconds each. This infinite loop, which was also used in task two, uses a Boolean called infinite which is set to true in the setup code. The following code is used in the loop body to enter the infinite loop:

while(infinite == true)

{

//two codes that loop for ten seconds each forever

}

Once the light sensor has been calibrated there is not much to this program other than copying and pasting the code from each of the previous tasks, but as there is no time-based loop in Arduino, looping each for ten seconds was the next problem to solve. This problem was solved by using the delays in each of the programs and using a loop to make each one last for 10 seconds. For instance, in the temperature sensor program there is a delay for 1.25 seconds between each temperature read, so if that block of code is ran 8 times, it will have run for 10 seconds. This was accomplished using a for loop.

After each program is ran for ten seconds, the program clears the screen and runs the next program for ten seconds and repeats the two codes forever.

# Results

## Rising/falling temperature sensor

The results from this first task were mostly positive with few programming errors. The biggest error that was encountered was having variables switched, for instance the average variables for the new and old average. With these two variables switched it messed up the comparison of the two averages. Once that was fixed, the average comparisons were accurate.

The next challenge that presented itself was displaying the words “falling”, “rising” or “stable” on different segments of the LCD screen. Arduino is limited, and it is not possible to use a command to clear only one row or the other on the LCD. Because of this, if one average comparison was rising and the next was falling, because each of those words were displayed on opposite sides of the LCD, both words would be displayed at once. This problem was solved by printing spaces on all the unused character spots when printing each word. For example, to display the word falling on the right side, the following commands were used:

lcd.setCursor(0,1);

lcd.print(" rising");

The spaces before the word rising would replace any words or characters displayed before the word with an empty space.

Other than those two challenges, the first task was carried out successfully with no other errors.

## Light calibration system

Unlike the first task, the second task presented many issues ranging from inconsistent hardware to programmer error and misconceptions.

The first error presented was getting the values backwards for the calibration of the photoresistor. The team failed to recognize that the more light the photoresistor is exposed to, the higher the raw value, the team thought of it as the reverse. After some troubleshooting and looking at the serial monitor, the values were switched, and the calibration worked smoothly.

The other challenge that presented itself was forgetting to clear the previous brightness level from the screen before printing the next one. Because of this, if the sensor read the brightness as 100 (a three-digit value), any following brightness reading that were two digits, would keep that third digit on the screen. In order to solve this issue, the three potential spaces for the percentage to be displayed were cleared before displaying the new percentage using the following code:

lcd.setCursor(6,1);

lcd.print(" ");

lcd.setCursor(6,1);

lcd.print(brightness);

Other than these two programming issues, which took some time to troubleshoot and solve, the only other problem encountered was the inconsistencies in the photoresistor itself. The button would get pressed two times under the same conditions and would provide two values on opposite sides of the spectrum. To remedy this issue, the team had the serial monitor pulled up the entire time and if the photoresistor gave a result that was not what it should have given, for example giving a very bright value when being covered, the board was reset.

## Alternating between temperature sensor and light value

Of the three tasks, this task was by far the easiest one to put together. The bulk of this task was copying and pasting the previous two tasks into one big program. Despite this simplicity, there were still a couple of minor issues that were encountered.

The first of these issues being trying to find a way to loop something for a certain period with no time-based loop in Arduino. After some brainstorming, the team noticed that it was possible to take advantage of the preexisting delays in each program to have it loop for ten seconds. For example, the temperature sensor program had a built-in delay of 1.25 seconds, if that block was looped 8 times, it would have run for 10 seconds.

The second issue was finding exactly where to put each of the lcd.clear() statements in the program so that none of the text from each individual program would overlap. Solving this took nothing more than running the program a couple of times, each time seeing where it ran into overlap and putting a clear statement there.

Other than these two issues, the program was no more complicated than copying and pasting the functions in the correct spots.

# Conclusion

This lab was a very good learning experience, each task presented its own specific challenges and learning experiences.

The first task taught a lot about clearing the LCD and that there is no command to only clear one row or column on the LCD. In order to only clear one row or column, spaces must be printed in place of the characters. The first lab also taught the importance of order in code. Specifically, the order in which variables are initialized. In future labs the team will pay close attention to the specific order in which the variables are initialized, and values are assigned.

The second task taught about how the photoresistor works. Any future labs that use the photoresistor the team will make sure that the values are being read correctly with darker values being higher.

While the third program was the easiest of the three, it taught a lot about thinking outside of the box with programming. With the third program, the team had to figure out a creative way to have each code loop for 10 seconds without a time-oriented loop.

# **Biographies**

**Keaton Raymond** is a first-year student at the University of Colorado at Colorado Springs. He was born and raised in Colorado Springs and graduated high school from Discovery Canyon Campus. Keaton has been interested in computers, programming, and engineering from a very young age. This influenced his path in pursuing a Computer Engineering major because it blends the hardware and software of computers. In high school, Keaton discovered a passion for physics which ultimately geared him towards a minor in Physics.

**Amyleila Mejia** graduated from Benjamin Franklin High School located in Los Angeles, California. She joined the military at the age of twenty and discharged honorably at the age of twenty-three. After serving only three years her military experience led Ms. Mejia into a very profound interest in all things mechanical and technical. She is currently enrolled as a student in the school of Electrical Engineering at the University of Colorado at Colorado Springs. Ms. Mejia hopes to make an impactful difference in society by contributing to the education of younger generations, specifically, in the fields of Science and Mathematics.

Appendix

//

// Lab11\_Task1

//

//

//Name: Keaton Raymond and Amyleila Mejia

//Date:11/5/2019

//Class: ECE 1001-005

//

//

//Problem Statement:

//display 3 temperatures from newest to oldest with the newest being on the left

//if the average of the three temps is .15 deg higher compared to the previous avg

//display rising on the right side of the bottom row

//if the average is .15 deg lower compared to the previous avg display falling

//on the left side of the bottom row

//if it is within .15 of the previous avg display stable in the middle

//delay 1.25 sec between each temp read

//

//

// ^^^^ Algorithm ^^^^

//

// define and initialize variables

// float voltage, tempRead, temp1-3, old avg, new avg

// initialize lcd

//

// setup function

// setup lcd

// pinMode(A0, INPUT) (temp sensor)

//

// loop function

// read voltage and convert to deg celsius

// temp3 = temp2

// temp2 = temp1

// temp1 = tempRead

//

// print the three temperatures with temp1 being on the left and temp3 being on the right

//

// calculate new average

// if statements comparing if the old avg is higher than the new avg and by how much

// and what to display on the bottom row of the lcd

//

// oldAvg = newAvg

// delay(1250)

//

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

float voltage = 0.0;

float tempRead = 0.0;

float temp1 = 0.0; //left temp

float temp2 = 0.0; //middle temp

float temp3 = 0.0; //right temp

float oldAvg = 0.0;

float newAvg = 0.0;

void setup()

{

lcd.begin(16, 2);

lcd.clear();

pinMode(A0, INPUT);

} //end setup

void loop()

{

//read the temp, convert to celsius

voltage = analogRead(A0) \* 4.8828125;

tempRead = (voltage - 500)/10;

//move old temps over, new temp on the left

temp3 = temp2;

temp2 = temp1;

temp1 = tempRead;

//display each temp in the appropriate position

lcd.setCursor(0,0);

lcd.print(temp1);

lcd.setCursor(6,0);

lcd.print(temp2);

lcd.setCursor(12,0);

lcd.print(temp3);

//calculate average temp

newAvg = (temp1+temp2+temp3)/3;

//if the new avg is at least .15 deg less than the previous avg, display falling on the left

if(oldAvg - newAvg >= 0.15)

{

lcd.setCursor(0,1);

lcd.print("falling ");

} //end if new avg is lower

//if the new avg is at least .15 deg more than the previous avg, display rising on the right

else if(newAvg - oldAvg >= 0.15)

{

lcd.setCursor(0,1);

lcd.print(" rising");

} //end of new avg is higher

//if the new avg is within .15 deg than the previous avg, display stable in the middle

else

{

lcd.setCursor(0,1);

lcd.print(" stable ");

} //end if new avg is within 0.15

oldAvg = newAvg; //make the new avg the old avg so you can store it as the last average

delay(1250);

} //end loop function

//

// Lab11\_Task2

//

//

//Name: Keaton Raymond and Amyleila Mejia

//Date:11/5/2019

//Class: ECE 1001-005

//

//

//Problem Statement:

//have the user calibrate a photoresistor using a button

//only run the calibration code one time

//then infinitely tell the user what percentage of the maximum brightness

//it is reading, using the map() command

//

//

// ^^^^ Algorithm ^^^^

//

// define and initialize variables

// initialize lcd

// define button 5

// int fromHigh, fromLow, brightness

// bool infinite = true

//

//

// setup function

// setup lcd

// button input pullup

// A1 input

//

// loop function

// tell user to press button in the dark

// wait for user to press button and record that light value as fromLow

// delay(500)

// tell user to press button in bright and record that light value as fromHigh

//

// enter infinite loop

// map and constrain brightness to 100

//

//

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

#define button 5

int fromHigh;

int fromLow;

int brightness;

bool infinite = true; //bool for infinite loop

void setup()

{

lcd.begin(16, 2);

lcd.clear();

pinMode(button, INPUT\_PULLUP);

pinMode(A1, INPUT);

Serial.begin(9600);

}

void loop()

{

//ask user to press button in dark

lcd.setCursor(0,0);

lcd.print("Press button in");

lcd.setCursor(0,1);

lcd.print("the dark");

do

{

fromLow = analogRead(A1);

} while(digitalRead(button) == HIGH); //end darkness do while loop

Serial.println(fromLow);

//ask user to press button in the light

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Press button in");

lcd.setCursor(0,1);

lcd.print("brightest area");

delay(500); //500 ms debounce

do

{

fromHigh = analogRead(A1);

} while(digitalRead(button) == HIGH); //end light do while loop

Serial.println(fromHigh);

//print text to tell people the brightness

lcd.clear();

lcd.setCursor(0,0);

lcd.print("The brightness");

lcd.setCursor(0,1);

lcd.print("is at");

while(infinite == true)

{

//read brightness and map it to a percentage

brightness = analogRead(A1);

brightness = map(brightness, fromHigh, fromLow, 0, 100);

brightness = constrain(brightness, 0, 100);

//clear last number and put a "%" sign

lcd.setCursor(6,1);

lcd.print(" ");

lcd.setCursor(6,1);

lcd.print(brightness);

lcd.setCursor(9,1);

lcd.print("%");

delay(750);

Serial.println(brightness);

} //end infinite loop

} //end loop function

//

// Lab11\_Task3

//

//

//Name: Keaton Raymond and Amyleila Mejia

//Date:11/5/2019

//Class: ECE 1001-005

//

//

//Problem Statement:

//have the user calibrate a photoresistor using a button

//only run the calibration code one time

//then infinitely tell the user what percentage of the maximum brightness

//it is reading, using the map() command for 10 seconds

//then repeat the temperature code for 10 seconds

//infinitely loop the brightness and temperature code

//

//

// ^^^^ Algorithm ^^^^

//

// define and initialize variables

// initialize lcd

// define button 5

// int fromHigh, fromLow, brightness

// float voltage, tempRead, temp1-3, old avg, new avg

// bool infinite = true

//

//

// setup function

// setup lcd

// button input pullup

// A1 input (photoresistor)

// pinMode(A0, INPUT) (temp sensor)

//

// loop function

// tell user to press button in the dark

// wait for user to press button and record that light value as fromLow

// delay(500)

// tell user to press button in bright and record that light value as fromHigh

//

// enter infinite loop

// map and constrain brightness to 100

// delay(750)

// loop for 10 seconds

//

// read voltage and convert to deg celsius

// temp3 = temp2

// temp2 = temp1

// temp1 = tempRead

//

// print the three temperatures with temp1 being on the left and temp3 being on the right

//

// calculate new average

// if statements comparing if the old avg is higher than the new avg and by how much

// and what to display on the bottom row of the lcd

//

// oldAvg = newAvg

// delay(1250)

// repeat for 10 seconds

//

//

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

#define button 5

int fromHigh;

int fromLow;

int brightness;

bool infinite = true; //bool for infinite loop

float voltage = 0.0;

float tempRead = 0.0;

float temp1 = 0.0; //left temp

float temp2 = 0.0; //middle temp

float temp3 = 0.0; //right temp

float oldAvg = 0.0;

float newAvg = 0.0;

void setup()

{

lcd.begin(16, 2);

lcd.clear();

pinMode(button, INPUT\_PULLUP);

pinMode(A0, INPUT);

pinMode(A1, INPUT);

Serial.begin(9600);

} //end setup

void loop()

{

//ask user to press button in dark

lcd.setCursor(0,0);

lcd.print("Press button in");

lcd.setCursor(0,1);

lcd.print("the dark");

do

{

fromLow = analogRead(A1);

} while(digitalRead(button) == HIGH); //end darkness do while loop

Serial.println(fromLow);

//ask user to press button in the light

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Press button in");

lcd.setCursor(0,1);

lcd.print("brightest area");

delay(500); //500 ms debounce

do

{

fromHigh = analogRead(A1);

} while(digitalRead(button) == HIGH); //end light do while loop

Serial.println(fromHigh);

while(infinite == true)

{

for(int i=0; i<13; i++)

{

//print text to tell people the brightness

lcd.clear();

lcd.setCursor(0,0);

lcd.print("The brightness");

lcd.setCursor(0,1);

lcd.print("is at");

//read brightness and map it to a percentage

brightness = analogRead(A1);

brightness = map(brightness, fromHigh, fromLow, 0, 100);

brightness = constrain(brightness, 0, 100);

//clear last number and put a "%" sign

lcd.setCursor(6,1);

lcd.print(" ");

lcd.setCursor(6,1);

lcd.print(brightness);

lcd.setCursor(9,1);

lcd.print("%");

delay(750);

Serial.println(brightness);

} //end 10 second brightness loop

lcd.clear(); //clear text from brightness loop

for(int j=0; j<8; j++) {

//read the temp, convert to celsius

voltage = analogRead(A0) \* 4.8828125;

tempRead = (voltage - 500)/10;

//move old temps over, new temp on the left

temp3 = temp2;

temp2 = temp1;

temp1 = tempRead;

//display each temp in the appropriate position

lcd.setCursor(0,0);

lcd.print(temp1);

lcd.setCursor(6,0);

lcd.print(temp2);

lcd.setCursor(12,0);

lcd.print(temp3);

//calculate average temp

newAvg = (temp1+temp2+temp3)/3;

//if the new avg is at least .15 deg less than the previous avg, display falling on the left

if(oldAvg - newAvg >= 0.15)

{

lcd.setCursor(0,1);

lcd.print("falling ");

} //end if new avg is less than old avg

//if the new avg is at least .15 deg more than the previous avg, display rising on the right

else if(newAvg - oldAvg >= 0.15)

{

lcd.setCursor(0,1);

lcd.print(" rising");

} //end if new avg is more than the old avg

//if the new avg is within .15 deg than the previous avg, display stable in the middle

else

{

lcd.setCursor(0,1);

lcd.print(" stable ");

} //end else new avg is within .15 deg of old avg

oldAvg = newAvg; //make the new avg the old avg so you can store it as the last average

delay(1250);

} //end 10 second brightness loop

lcd.clear(); //clear temperature text

} //end infinite loop

} //end loop

1. [↑](#footnote-ref-1)