To: Jack Garrard
From: Group 1: Fernando Garcia, Nikhil Frattare, Tlaloc Xiloyotl
Date: September 30, 2024
Subject: P1 - Deliverable 5

General Introduction:

General Introduction:

This report provides documentation for the design process and creation of a temperature sensor circuit including the program code required to run it. This includes how the TMP36 was used in order to achieve all requirements set by the client. The following attachments are also included in this report:

I. Pairwise Matrix
II. Gantt Chart(s)
III. Flow Chart
IV. Arduino Code
V. Schematic(s)

Introduction and problem statement:

For this project, the customer desired an electronic thermometer. Within this report, our decision process will be outlined and explained. This includes project requirements, project restraints, background research, and our decision matrix. It then goes on to explain how we built and tested the product till it reached its final form. Afterwards, there is a discussion on failures and future improvements.

Project requirements and constraints:

Specific Requirements

- Minimum thermometer range: -10° to 150°F
- Minimum thermometer accuracy: ±2C°
- Minimum 4 LEDs used within the circuit
- Visual output in Celsius and Fahrenheit
- Switch must be integrated into Arduino software

General Requirements

- Must be able to measure general room temperature
- Must be relatively small
- Must have some visual display of the temperature

Constraints

- Must use one of the following temperature sensors:
  - TMP36
  - LM34
  - LM334
  - AD 592
  - TMP37
  - LM61
- Must use a microcontroller
- Must utilize a switch in some manner

Background Research:

When researching the available transducers we were allowed to use, we made a set of qualities that we would be looking for. This included, but was not limited to qualities such as accuracy at room temperature, maximum inaccuracy, voltage output, and maximum temperature range. Most transducers had accuracy within our requirements, and most had the correct temperature range. However we noticed that both the AD592 and LM334 output a current instead of a voltage, and we deemed them to be harder to work with.

Decision Matrix:

Criteria	Weights	TMP36	LM34	LM334	AD592	TMP37	LM61	Sums
Temperature Range	0.11600	0.22673	0.23031	0.11933	0.15513	0.11337	0.15513	1.00000
Input Voltage Range	0.09875	0.30110	0.07054	0.06022	0.07262	0.30110	0.19441	1.00000
Maximum Error	0.24004	0.15914	0.09823	0.10609	0.31827	0.15914	0.15914	1.00000
Cost	0.17083	0.18319	0.04259	0.35192	0.03386	0.17950	0.20895	1.00000
Long Term Stability (Lifespan)	0.19856	0.14286	0.14286	0.14286	0.28571	0.14286	0.14286	1.00000
Intended Environment Accuracy	0.17582	0.19149	0.10638	0.06383	0.38298	0.19149	0.06383	1.00000
Score	1.00000	0.18756	0.11161	0.14496	0.23141	0.17378	0.15068	1

Figure 1: Decision Matrix

Using the pairwise matrix (**Figure 2**), we calculated weights for each criteria. The most important weights came out to be the maximum error, lifespan of the sensor, and the accuracy in the intended environment (room temperature). In general, the AD592 scored the best for the categories worth the most, however it scored lower in cost and temperature range than other sensors. Overall, the AD592 performed the best and came out to be the best choice within the decision matrix. In close second, the TMP36 performed similarly. It outperformed in the temperature range category, but fell shorter when it came to accuracy and lifespan. In the end, we decided on the TMP36 because it was much easier to work with than the AD592, was closely behind it in the matrix, and still met all the project requirements.

Building and Testing:

Initially, we first wanted to get any output from the TMP36. At first, we almost destroyed the transducer by confusing the GND and Vin pins. Once figuring those out, we wrote very basic code to get the analog reading pin working with the sensor. After getting a reading, we did some research on how the Arduino reads the sensor. Using the information, we were able to reverse the reading process and get actual voltage readings from the sensor. Then, using the data sheet provided for the sensor, we could

find the temperature in Celsius based on the voltage reading. After getting the temperature reading, we decided on how we wanted to take care of inaccurate extraneous results. Our decision was to take the results multiple times a second, and average them, displaying the average approximately every second. When implementing that, it succeeded the first time, and we tried varying numbers until we had an accurate and stable reading. We then attempted to implement the LCD into the circuit. When implementing it, we ran into an issue with the breadboard we chose being faulty. We then had to replace the breadboard and move all the components to a new one. After some trial and error, along with more research on specifically implementing it with an arduino, we got the LCD to display the text of the temperature. Next, we decided to get the button working to change the display on the LCD. We went through multiple versions of the code until we got one that was efficient and worked. We ended up creating two separate outputs, and toggled which one was displayed based on if the button was pressed upon refresh of the display. The button was integrated into the circuit with very few issues. Then, we had to decide what information the LEDs would display. Initially, we discussed if they should be simple tasks such as turning one on when the device is working. After realizing we had multiple colors of LEDs, we decided to use three of them for a temperature range. The red would turn on when it is “hot”, the blue for “cold”, and the green for “room temperature”. The final LED ended up being used for every refresh of the display, it would flash. Initially, due to the small size of the breadboard, we had a hard time making the LEDs work with resistors properly. After realizing the LEDs didn’t burn out without resistors, we solved a lot of our spacing issues with the breadboard. Overall, the project went through many iterations, mainly with the code being redone and cleaned up multiple times, and the circuit being rebuilt a few times.

Demo Results:

The project ended up with minor issues with potential cross-wiring, or lack of resistors, and a mistake in one of the lines of code pertaining to the refresh led. Some potential improvements that could be made to the device could be more organized cable management as well as potentially adding a resistor to the LEDs to constrain the brightness as well as stop any leaking of voltage to affect the other LEDs in the vicinity. A simple fix that we could immediately implement would be to fix our code for the refresh LED as we have an invalid parameter for it. We would have also liked to clean up the code for the LEDs as it was a lot of if and else statements.

Design Process Discussion:

1. Objectives and Criteria
    - At the beginning of the project, we were given a set of transducers.
    - As a team, we created a decision and pair matrix to decide on what transducer would be best for how we wanted to implement the transducer.
    - After a couple of days of research. We ended up settling for the TMP36

- Later in the project, we were given requirements and specifications by the customer that we had to abide by and account for when designing the temperature sensor.
- Since we chose the TMP36 our one and only problem we had to solve in order to convert mV to Celsius.
- Closer to the end of the project, we were required to make a Gantt chart to highlight our road to the final product and to keep track of how we spent our time, and to see where we could improve our time management.

2. Analysis and Synthesis
    - We began to generate the concept of what we wanted to build through more research and seeing examples of our transducer and LCD display being used in similar environments.
    - We divided the project into subsections as we went on.
    - We first focused on making the transducer work and read temperatures that were accurate and readable to our final consumer.
    - We then focused on the LCD display and evaluated how to make it as beneficial to the customer. We made the LCD's display saturation adjustable so that if someone can't see the temperature with the default settings they can adjust it to their liking.
    - After the LCD Display, we focused on the button requirement. We decided to use the button to toggle between Celsius and Fahrenheit to adjust to the final customer's preference.
    - Finally, we capped off the project by implementing the 4 LEDs requirement. We decided to use the LEDs for simple, but helpful implementations.
3. Construction
    - We worked on the device for about a week to 2 weeks where we figured out how to wire all the components properly as well as code everything to work properly.
    - First, we coded and wired the TMP36 so that we got Fahrenheit values in the Arduino IDE through the known Celsius to Fahrenheit conversion and implemented the formula to change the voltage being read by the arduino to temperature.
    - Somewhere in between the first and second steps we implemented a code so that the display of the temperatures was not as fast using a method to average the temperatures every 100 readings of temperatures.
    - Then, we coded and wired the LCD 1602 screen in a way that displayed the temperatures as displayed on the Arduino Serial Monitor.
    - After the LCD screen implementation, we made the code that allowed for the usage of the switch. The code allowed for the switch to be used as a toggle switch that switched from C to F or vice versa.
    - The final step of the construction process was to implement the 4 LEDs required by the customer. We decided to have 3 LEDs work as temperature indicators. Blue for cold

(temperature < -17 C), red for hot (temperature > 37 C), and green for room temperature (20 C - 30 C). The final LED was white and its purpose was to indicate when the LCD was refreshed. It did so by blinking every time the Arduino read new data.

4. Test and Evaluation
    - The final test consisted of high-temperature and very low-temperature tests. Our temperature sensor passed both the high and low temperature tests.
    - Our switch worked and passed the requirement test.
    - We had 4 functioning LEDs but unfortunately, we had an issue with the dim blinking of our green LED due to an unknown error.

Appendix I:

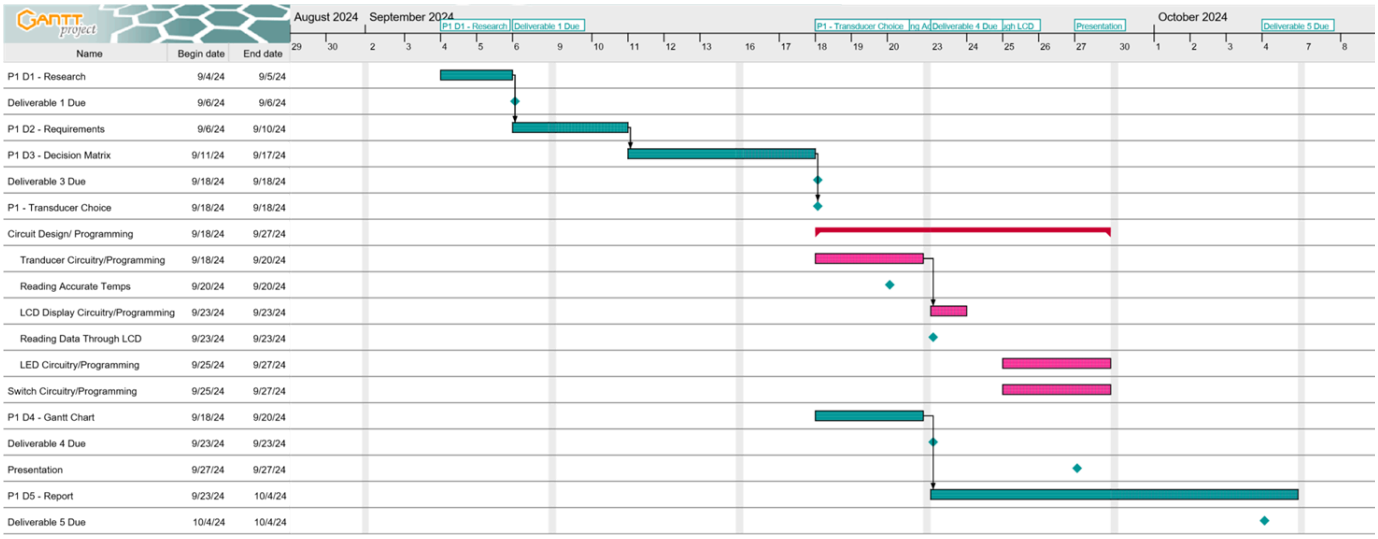
	Temperature Range	Voltage Range	Maximum Error	Cost	Long Term Stability (Lifespan)	Intended Environment Accuracy
Temperature Range	1.00	1.00	0.33	0.50	0.50	0.50
Input Voltage Range	1.00	1.00	0.20	0.25	0.33	0.50
Maximum Error	3.00	5.00	1.00	2.00	2.00	1.00
Cost	2.00	4.00	0.50	1.00	1.00	0.50
Long Term Stability (Lifespan)	2.00	3.00	0.50	1.00	1.00	3.00
Intended Environment Accuracy	2.00	2.00	1.00	2.00	0.33	1.00

Figure 2: Pairwise Matrix

Pairwise Matrix:

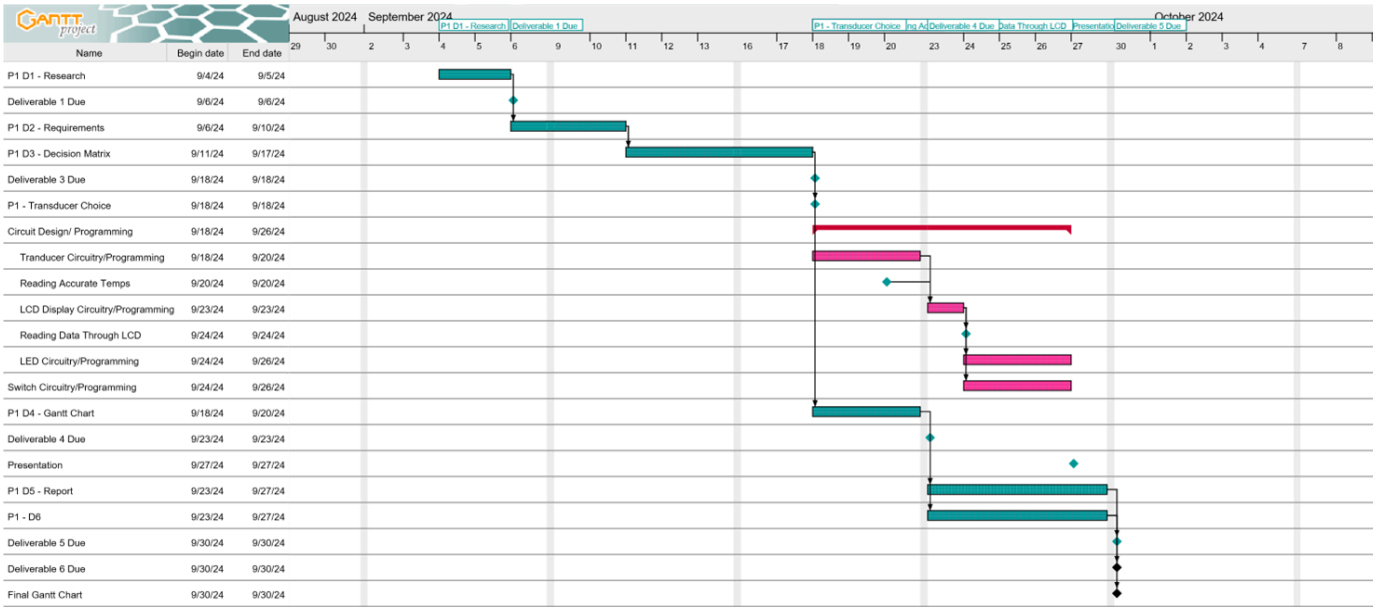
The pairwise matrix determined the weights given to each category in the decision matrix. Overall, the temperature range and voltage range categories were ranked the least important. Long term stability was considered more important than nearly every other category, and maximum error ended up being considered the most important.

Appendix II: Original and Final Gantt Chart(s)



Name	Begin date	End date
P1 D1 - Research	9/4/24	9/5/24
Deliverable 1 Due	9/6/24	9/6/24
P1 D2 - Requirements	9/6/24	9/10/24
P1 D3 - Decision Matrix	9/11/24	9/17/24
Deliverable 3 Due	9/18/24	9/18/24
P1 - Transducer Choice	9/18/24	9/18/24
Circuit Design/ Programming	9/18/24	9/27/24
Tranducer Circuitry/Programming	9/18/24	9/20/24
Reading Accurate Temps	9/20/24	9/20/24
LCD Display Circuitry/Programming	9/23/24	9/23/24
Reading Data Through LCD	9/23/24	9/23/24
LED Circuitry/Programming	9/25/24	9/27/24
Switch Circuitry/Programming	9/25/24	9/27/24
P1 D4 - Gantt Chart	9/18/24	9/20/24
Deliverable 4 Due	9/23/24	9/23/24
Presentation	9/27/24	9/27/24
P1 D5 - Report	9/23/24	10/4/24
Deliverable 5 Due	10/4/24	10/4/24

Figure 3: Original Gantt Chart



Name	Begin date	End date
P1 D1 - Research	9/4/24	9/5/24
Deliverable 1 Due	9/6/24	9/6/24
P1 D2 - Requirements	9/6/24	9/10/24
P1 D3 - Decision Matrix	9/11/24	9/17/24
Deliverable 3 Due	9/18/24	9/18/24
P1 - Transducer Choice	9/18/24	9/18/24
Circuit Design/ Programming	9/18/24	9/26/24
Tranducer Circuitry/Programming	9/18/24	9/20/24
Reading Accurate Temps	9/20/24	9/20/24
LCD Display Circuitry/Programming	9/23/24	9/23/24
Reading Data Through LCD	9/24/24	9/24/24
LED Circuitry/Programming	9/24/24	9/26/24
Switch Circuitry/Programming	9/24/24	9/26/24
P1 D4 - Gantt Chart	9/18/24	9/20/24
Deliverable 4 Due	9/23/24	9/23/24
Presentation	9/27/24	9/27/24
P1 D5 - Report	9/23/24	9/27/24
P1 - D6	9/23/24	9/27/24
Deliverable 5 Due	9/30/24	9/30/24
Deliverable 6 Due	9/30/24	9/30/24
Final Gantt Chart	9/30/24	9/30/24

Figure 4: Final Gantt Chart

Gantt Charts:

Our final Gantt chart was more connected through predecessors from start to finish compared to our second Gantt chart and in our final Gantt chart, there are also more deliverables and due dates listed on the chart as the date when it was made was after all parts of the project were complete.

Appendix III:

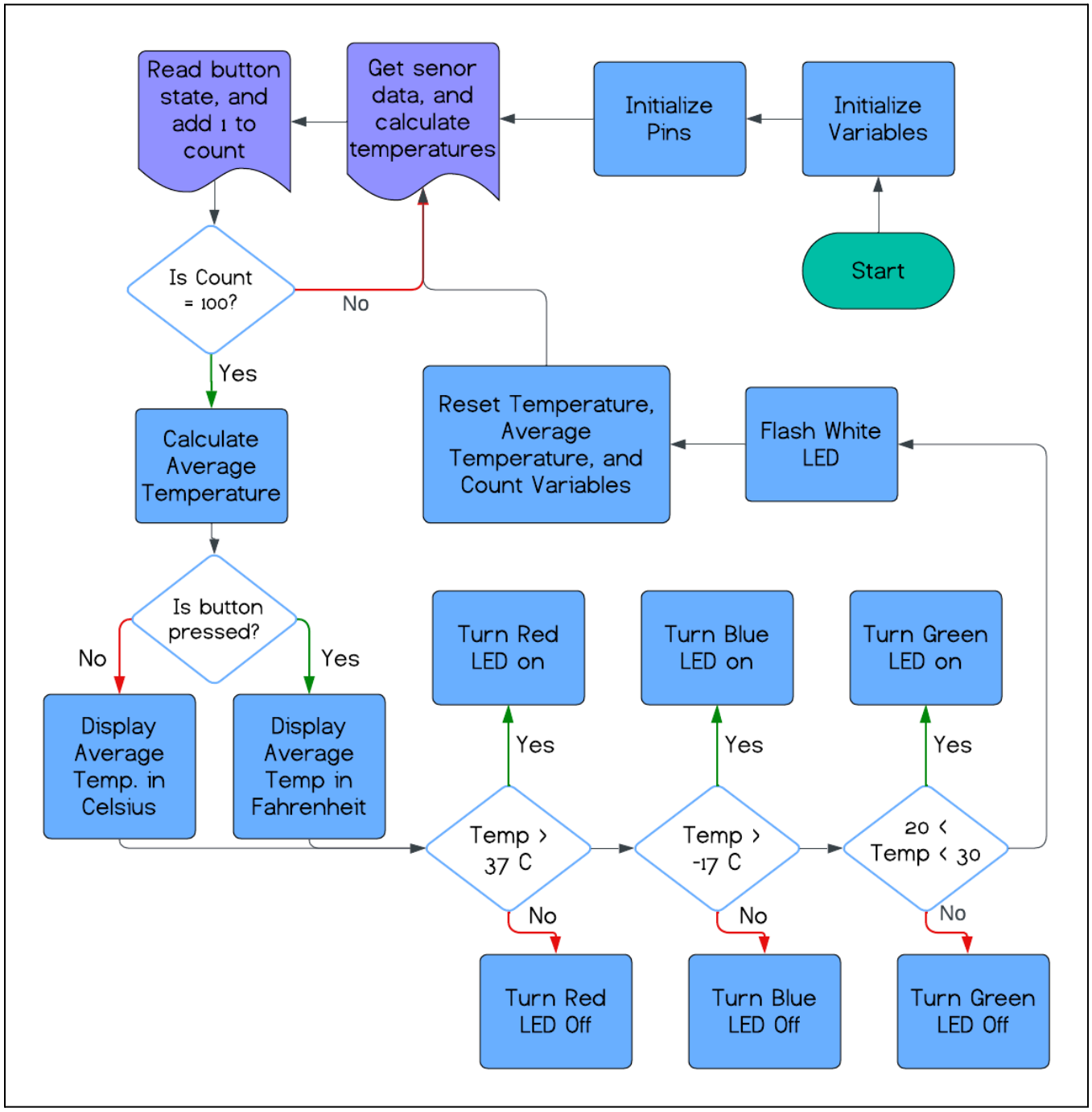


Figure 5: Flow Chart

Appendix IV: Arduino Code

```
#include <LiquidCrystal.h>      //includes the library for the LCD
const int temp_sensor = A0;     //initialize the temp sensor through pin A0
const int button = 7;           //initialize the button through pin 7
const int led_hot = 8;          //initialize the red LED through pin 8
const int led_cold = 9;         //initialize the blue LED through pin 9
const int led_refresh = 6;      //initialize the white LED through pin 6
const int led_roomtemp = 10;    //initialize the green LED through pin 10

float temp = 0;       //create the temperature variable
float avg_temp = 0;   //create the variable for average temperature
int count = 0;        //create the count variable
bool toggle = 0;      //create the button toggle variable
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);    //initialize the LCD through multiple pins

void setup() {
  // put your setup code here, to run once:
  lcd.begin(16, 2);   //start the LCD to display 16x2
  Serial.begin(9600);   //put serial out through 9600
  pinMode(button, INPUT);   //initialize the button pin to be an input
  pinMode(led_hot, OUTPUT);   //initialize the red LED pin to be an output
  pinMode(led_cold, OUTPUT);    //initialize the blue LED pin to be an output
  pinMode(led_roomtemp, OUTPUT);    //initialized the white LED pin to be an output
  pinMode(led_refresh, OUTPUT);   //initialized the green LED pin to be an output
}

void loop() {
  float input_v = analogRead(temp_sensor);   //read the temperature sensor and assign the reading to
variable "input v"
  float volts = input_v / 1023 * 5 - 0.5;    //take that input, and convert it into an actual voltage reading
by reversing the arduino input and offset of the sensor
  float C = volts *100;    //multiply by 1000 and divide by 10 (multiply by 100) because the sensor reads
10mV per degree Celsius

  bool button_state = digitalRead(button);   //read the button state and assign it to the "button state"
variable
  delay(10);    //delay the arduino by 10ms
  temp += C;    //add to the temp variable, the current reading temp in Celsius
  count += 1;   //add one to the count
```

```
 if (count == 100){
   float avg_temp = (temp / 100);   //averages the last 100 temperature readings (essentially readings
over 1 second)
   Serial.println((String) avg_temp + "°C");   //Serial prints the Celsius temperature
   Serial.println((String)(avg_temp * 1.8 + 32) + "°F");   //Serial prints the Fahrenheit temperature
   Serial.println("~~~~~~~~~~~~~~");    //Serial prints a spacer line
   lcd.clear();    //clears the LCD
   lcd.setCursor(0,1);   //Sets the LCD to start writing at the beginning

   if (button_state == 1){
   toggle = !toggle;   //If button is clicked, toggle the "toggle" variable to the opposite state
   }

   if (toggle == 1){
     lcd.print(avg_temp * 1.8 + 32, 1);  //If button toggle is True, outputs to the LCD the Fahrenheit
temperature
     lcd.print(char(223));   //Outputs the degrees symbol
     lcd.print("F");   //Outputs "F" for Fahrenheit
   }
   else{
     lcd.print(avg_temp);    //If button toggle is False, outputs to the LCD the Celsius temperature
     lcd.print(char(223));   //Outputs the degrees symbol
     lcd.print("C");   //Outputs "C" for Celsius
   }

   if (avg_temp > 37){     //If the average temperature reading is above 37, turns on the red LED
     digitalWrite(led_hot, HIGH);
   }
   else{     //Otherwise, turns off the "hot" LED
     digitalWrite(led_hot, LOW);
   }
   if (avg_temp < -17){    //If the average temperature reading is below -17, turns on the blue LED
     digitalWrite(led_cold, HIGH);
   }
   else{     //Otherwise, turns off the "cold" LED
     digitalWrite(led_cold, LOW);
   }
   if (count == 100){    //If count = 100, always true due to previous if, turns on the white LED for .1
seconds
```

```
    digitalWrite(led_refresh, HIGH);
    delay(count);
    digitalWrite(led_refresh, LOW);
   }
   if (20 < avg_temp && avg_temp < 30){    //If the average temperature reading is between 20 and 30,
turns on the green LED
    digitalWrite(led_roomtemp, HIGH);
   }
   if(avg_temp > 30){    //If average temperature more than 30, turns off green LED
    digitalWrite(led_roomtemp, LOW);
   }
   if(avg_temp < 20){    //If average temperature less than 20, turns off green LED
    digitalWrite(led_roomtemp, LOW);
   }
   avg_temp = 0;   //Resets "avg_temp" variable to  0
   temp = 0;   //Resets "temp" variable to 0
   count = 0;    //Resets "count" variable to 0
  }
}
```

Appendix V:

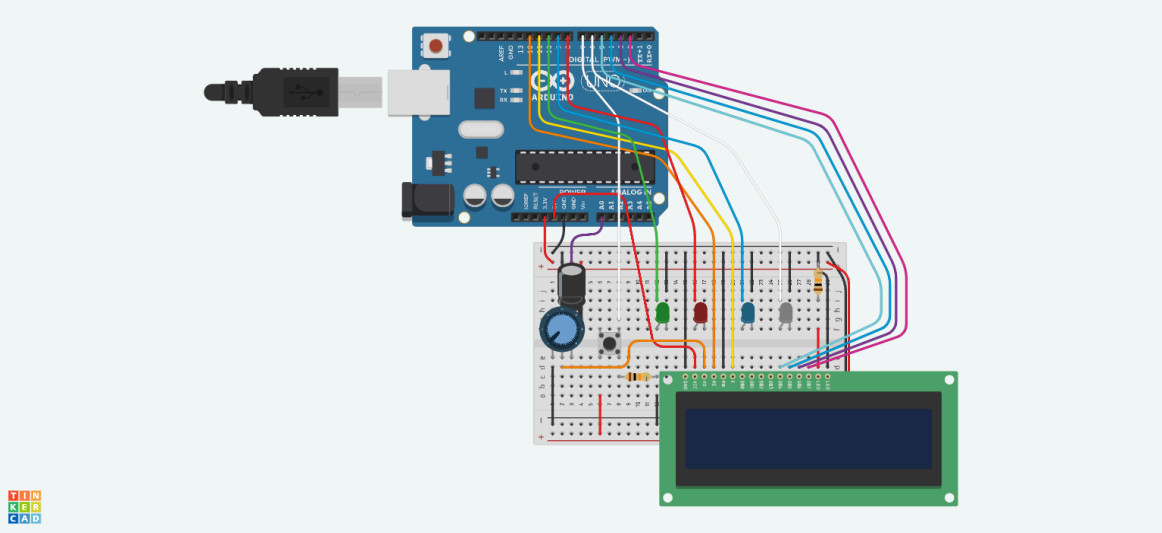


Figure 6: Circuit Schematic Visual

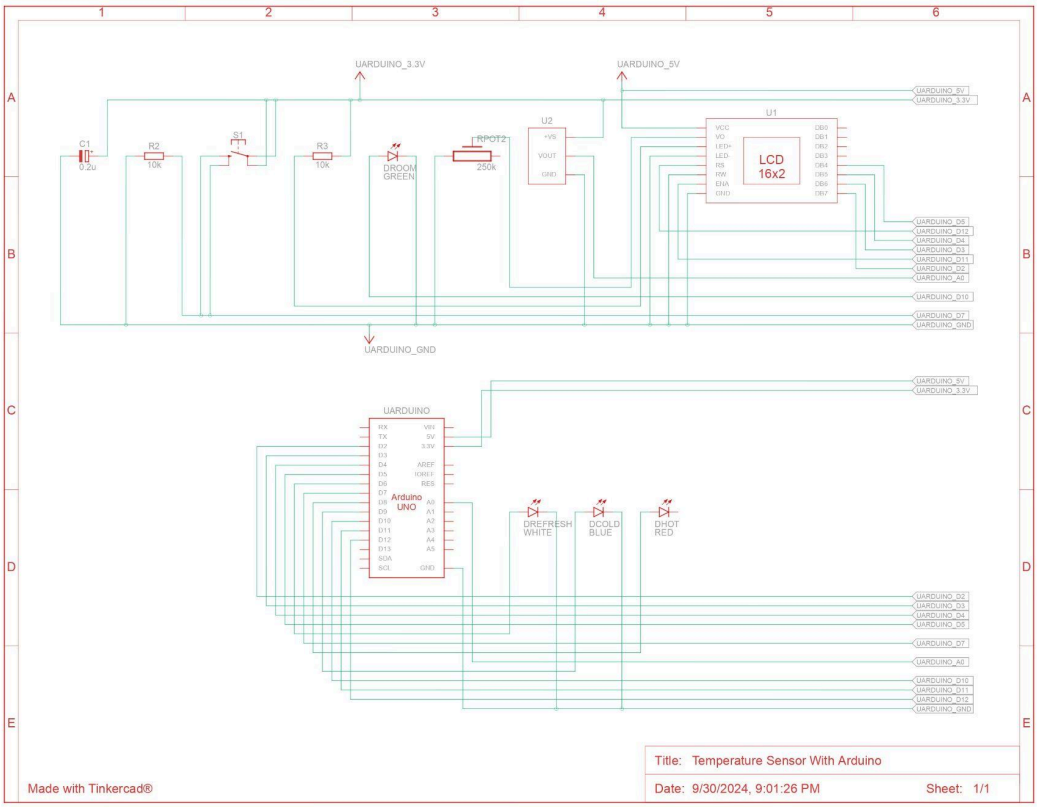


Figure 7: Circuit Schematic