**ELEC1601 Week 4 Lab Report**

SID: 510413811

Lab Group: R18\_Group13

Online student: Yes

**Introduction**

The purpose of this lab session was possibly to understand how to work with RGB LEDs, to familiarize with the advantages of multi-color signals, and to familiarize with the utilities of temperature sensors.

We understood the features and usages of the components mentioned above by reading the instructions on Canvas and googling. We successfully implemented the required circuits on TinkerCad by referring to the relevant pages from the Arduino Reference website, specifically the functions of analogWrite(), analogRead(), and digitalWrite().

**Materials**

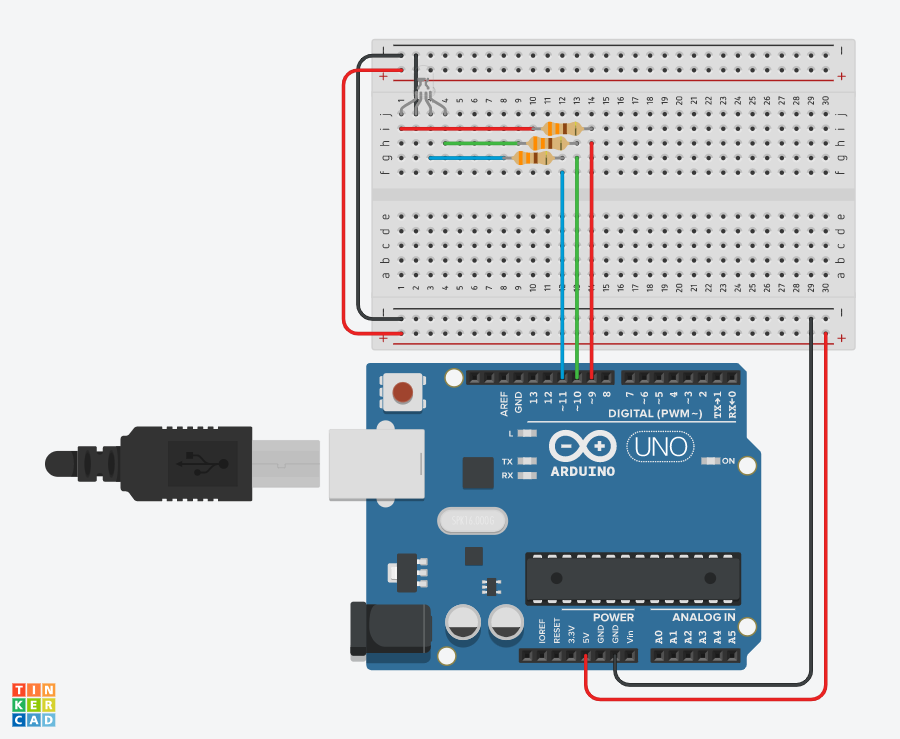
An RGB LED uses 3 LEDs of red, green, and blue to create a wide range of colors by controlling the brightness of each individual LED. “Once the power is in the correct direction to flow through the LED, the brightness is determined by the current or the speed of the flow of power. The more electrons filling holes, the more photons released, and the higher the brightness output (Diode Dynamics, 2021).”

A temperature sensor is a device that measures the surrounding temperature. “Unlike a thermistor, the TMP36 does not have a temperature-sensitive resistor. Instead, this sensor uses the property of diodes; as a diode changes temperature the voltage changes with it at a known rate. The sensor measures the small change and outputs an analog voltage between 0 and 1.75VDC based on it (BCRobotics, 2021).”

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element (ESComponents, 2017). Ohm’s Law indicates that current = voltage/resistance. Therefore, the purpose of any type of resistor is to mainly control the number of electrons going through part of the circuit where it is involved. By doing so it is possible to protect certain components or deliberately change the voltage output to reflect a particular environment.

**Connection**

**Task 1:**



*Pseudocode*

1. Assign red signal to pin 9, green signal to pin 10, blue signal to pin 11
2. Initialize pin9, pin10, and pin11 as OUTPUTs
3. Set red output to 255, green output to 0, blue output to 0 representing the color red
4. Delay for 500 ms
5. Set red output to 255, green output to 165, blue output to 0 representing the color orange
6. Delay for 500 ms
7. Set red output to 255, green output to 255, blue output to 0 representing the color yellow
8. Delay for 500 ms
9. Set red output to 0, green output to 255, blue output to 0 representing the color green
10. Delay for 500 ms
11. Set red output to 0, green output to 0, blue output to 255 representing the color blue
12. Delay for 500 ms
13. Set red output to 75, green output to 0, blue output to 255 representing the color indigo
14. Delay for 500 ms
15. Set red output to 143, green output to 0, blue output to 255 representing the color violet
16. Delay for 500 ms
17. Repeat from 3)

*Explanation*

* Communication speed between the computer and the board is set to 9600 bits per second
* The red, green, and blue wires are connected through a resistor of 330 Ohms to the red, green, blue signal pin of the LED correspondingly
* The red, green, blue signal pins are connected to pins 9, 10, 11 correspondingly
* The color of LED loops the sequence of red 🡪 orange 🡪 yellow 🡪 green 🡪 blue 🡪 indigo 🡪 violet

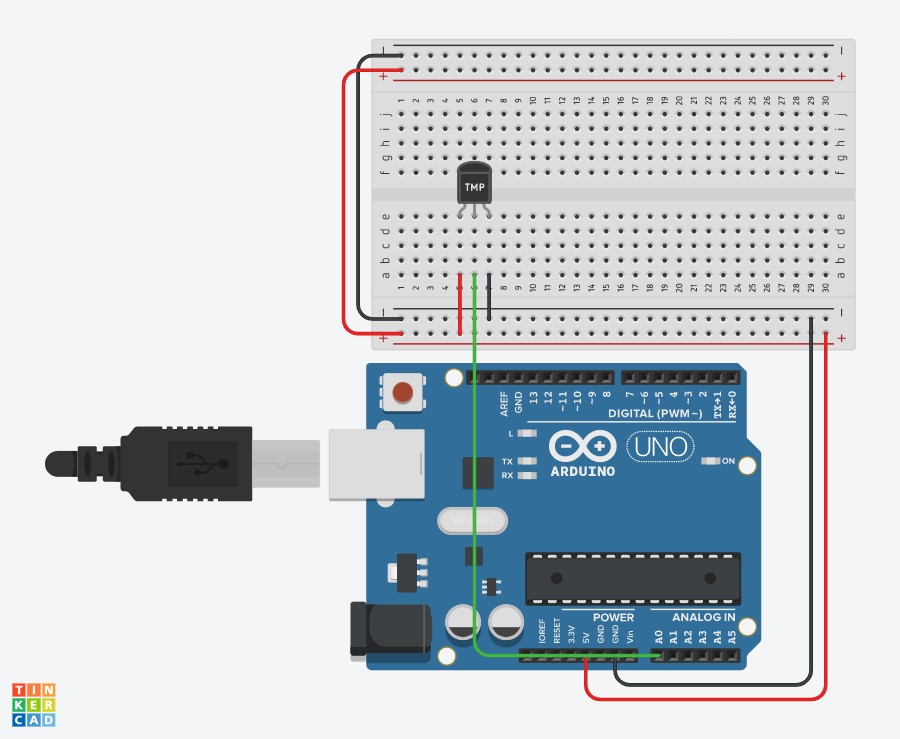
*Success*

* The code is straightforward
* The circuit connection is simple and easy to follow

*Limitation*

* We had to google for those RGB values of colors
* Although the color orange has a very different set of RGB values, it looks nearly identical to the color yellow on TinkerCad, which caused us a very long time doing unnecessary debugging

**Task 2:**



*Pseudocode*

1. Assign the thermometer to pin A0
2. Read the digitalized value from A0
3. Convert the digitalize value to the voltage measured
4. Convert the voltage measured to the degree in Celsius
5. Output degree in Celsius to the serial monitor

*Explanation*

* Communication speed between the computer and the board is set to 9600 bits per second
* The green wire connects the thermometer to the analog read pin A0
* The red wire connects the thermometer to the 5V power
* The black wire connects the thermometer to the ground
* The thermometer reads the virtual temperature of the environment and returns the digitalized value
* The value is then converted to the corresponding voltage measured and to the degree in Celsius
* After the final calculation, the degree in Celsius is printed on the serial monitor

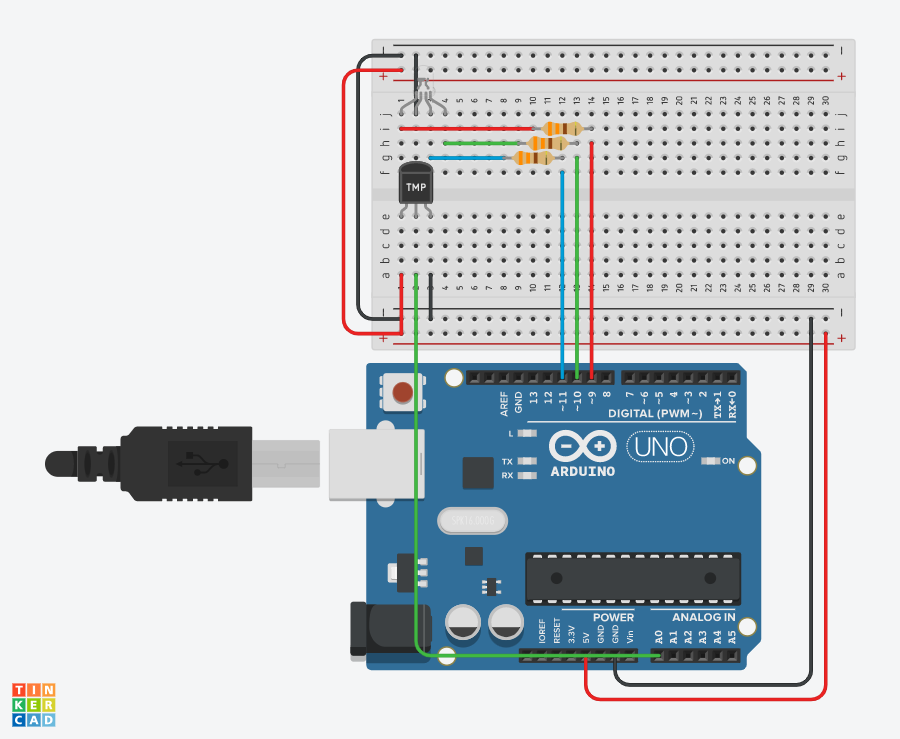
*Success*

* The code is straightforward
* The circuit connection is simple and easy to follow

*Limitation*

* We had to google the formula of converting digitalize value to the corresponding voltage measured for a given Arduino board component
* We had to find the offset of this particular model of temperature from the dataset document in order to convert voltage measured to the degree in Celsius

**Task 3:**



*Pseudocode*

1. Assign red signal to pin 9, green signal to pin 10, blue signal to pin 11
2. Initialize pin9, pin10, and pin11 as OUTPUTs
3. Assign the thermometer to pin A0
4. Read the digitalized value from A0
5. if the virtual temperature is (0, 25], the LED changes to blue
6. if the virtual temperature is (-40, 0], the LED changes to cyan
7. if the virtual temperature is (26, 50], the LED changes to yellow
8. if the virtual temperature is (50, 100], the LED changes to orange
9. if the virtual temperature is (100, 125], the LED changes to orange

*Explanation*

* Communication speed between the computer and the board is set to 9600 bits per second
* For the left section (thermometer):
  + The green wire connects the thermometer to the analog read pin A0
  + The red wire connects the thermometer to the 5V power
  + The black wire connects the thermometer to the ground
* For the right section (RGB LED):
  + The red, green, and blue wires are connected through a resistor of 330 Ohms to the red, green, blue signal pin of the LED correspondingly
  + The red, green, blue signal pins are connected to pins 9, 10, 11 correspondingly
* The thermometer reads the virtual temperature of the environment and returns the digitalized value
* The value is then converted to the corresponding voltage measured and to the degree in Celsius
* After the final calculation, The LED will change color based on the degree in Celsius read from the thermometer
  + If the temperature is below 0 (below water freezing point):
    - LED 🡪 cyan
  + If the temperature is between 0 and 25 (regular room temperature):
    - LED 🡪 blue
  + If the temperature is between 25 and 50 (below the highest recorded temperature in AUS):
    - LED 🡪 yellow
  + If the temperature is between 50 and 100 (below water boiling point):
    - LED 🡪 orange
  + If the temperature is above 100 (above water boiling point):
    - LED 🡪 red

*Success*

* Most of the code blocks are identical to the previous tasks
* The circuit connection is a simple combination of the previous tasks

*Limitation*

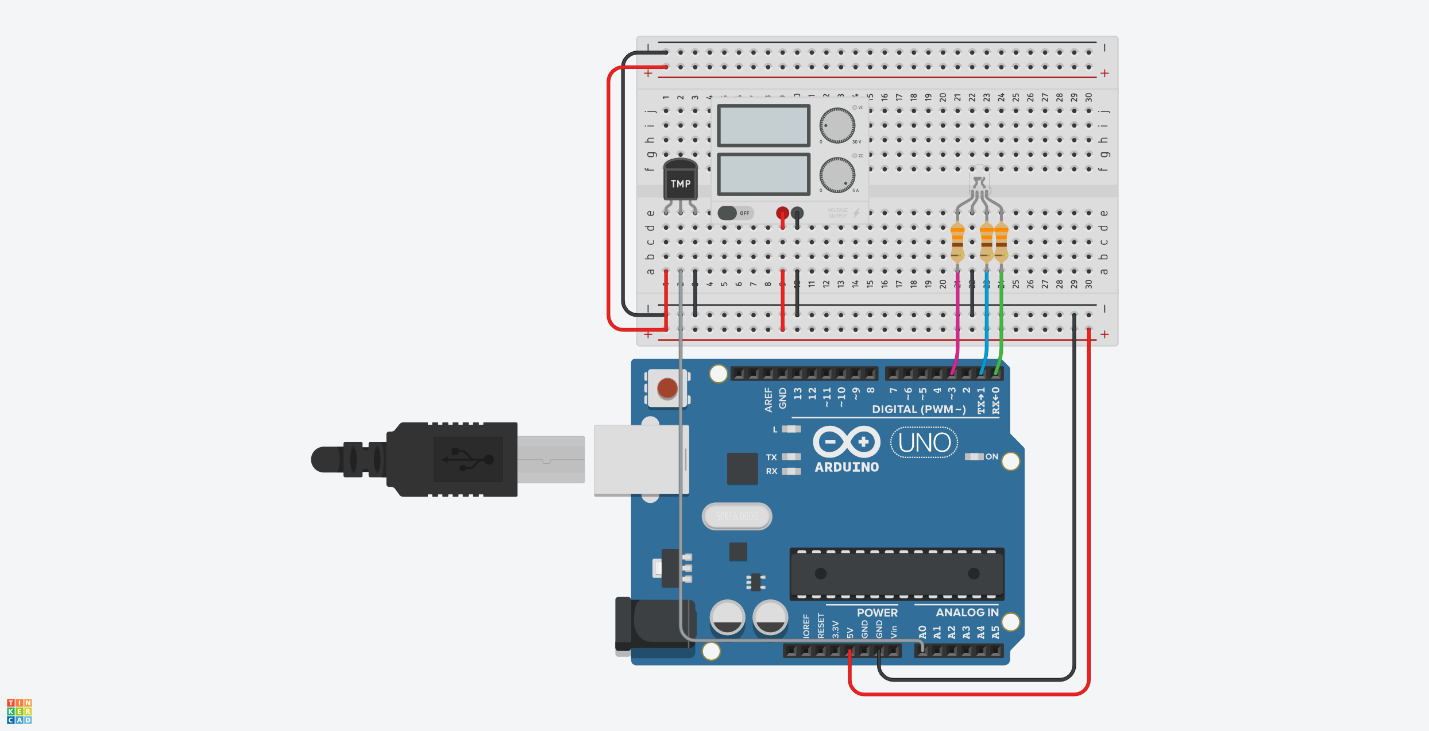
* We had to google for those RGB values of colors
* Although the color orange has a very different set of RGB values, it looks nearly identical to the color yellow on TinkerCad, which caused us a very long time doing unnecessary debugging
* We had to google the formula of converting digitalize value to the corresponding voltage measured for a given Arduino board component
* We had to find the offset of this particular model of temperature from the dataset document in order to convert voltage measured to the degree in Celsius

**Relation to real-world electronics**

*Description*

The setup of the combination of the LED display and a thermometer can be used to monitor the temperature of industrial machines such as generators. The color of the LED display can change correspondingly to the real-time temperature reading from the thermometer to inform the operator and can adapt the appropriate actions automatically. In the connection drawing below, I will use a power supply to represent the role of a temperature-monitored generator.

*Connections/Sensors*



*Pseudocode*

1. Assign red signal to pin 3, green signal to pin 1, blue signal to pin 0
2. Initialize pin0, pin1, and pin3 as OUTPUTs
3. Assign the thermometer to pin A0
4. Read the temperature in proximity of the servo motor from A0
5. if the temperature of the power supply is within normal range:
   1. the LED changes to green
6. if the temperature of the power supply is below the normal range:
   1. the LED changes to blue
   2. displays a warning message
7. if the temperature of the power supply is significantly below the normal range:
   1. the LED changes to cyan
   2. display a warning message
8. if the temperature of the power supply is above the normal range:
   1. the LED changes to yellow
   2. displays a warning message
9. if the temperature of the power supply is significantly above the normal range:
   1. the LED changes to red
   2. display a warning message

*Risks and Difficulties*

* When the program gets more complicated, multiple parts of the code can produce bugs that render down the effectiveness of the whole system including the following:
  + temperature reading
  + digitalized value converting
  + LED coloring changing
* Therefore, depends on the significance of overheating or overcooling, it may require more procedures other than simply a display and warning to ensure safety

References

Diode Dynamics. (2021). *LED brightness*. Retrieved from

https://www.diodedynamics.com/info/research/led/led-brightness.html

BCRobotics. (2021). *Using a TMP36 temperature sensor with Arduino.* Retrieved from

https://bc-robotics.com/tutorials/using-a-tmp36-temperature-sensor-with-arduino/

ESComponents. (2017). *Passive electronic components - Just what are they*. Retrieved from

https://www.escomponents.com/blog/2017/10/9/passive-electronic-components-just-what-are-they