UNIVERSITY OF CALGARY

SCHULICH School of Engineering

UNIVERSITY OF CALGARY

Rajdeep Singh

Index

Abstract

This project presents the development of an automated slag removal tool designed to enhance safety and efficiency in foundries and steel plants. Traditional slag removal techniques involve manual handling of molten materials, posing significant risks to workers. The proposed solution integrates mechanical components and an IoT-based control system. The tool effectively removes slag without physical contact with molten metal and also includes real-time temperature monitoring for hazardous environment awareness and safety for tool. This innovation promotes industrial safety, automation, and cost-efficiency.

1. Introduction

Slag removal is a critical process in foundries and steel industries to ensure high-quality casting and maintain equipment. Conventional methods often involve direct human intervention near molten metal, leading to high accident risks. Workers use metal rods to manually skim slag, which can cause burns, respiratory issues, and ergonomic stress.

The objective of this project is to create a smart, Radio Frequency-controlled slag removal device that eliminates the need for workers to come into close contact with molten material. This device combines mechanical tongs with Arduino-based control, integrating safety features like temperature sensors and visual-audio alerts. The result is a safe, reliable, and efficient industrial tool.

Relevance and Importance: The system directly addresses workplace safety challenges, reduces human error, and promotes automation. Its cost-effective nature also makes it suitable for small and medium-scale industries.

---

2. Research and Methodology

Literature Survey and Prior Work: Existing slag removal systems are largely manual or semi-mechanical, with little focus on automation and worker safety. While some industries have adopted automated cranes and ladle cleaners, they often lack affordability and adaptability for smaller operations.

Gap Analysis:

- High risk of burn injuries and accidents in traditional systems
- Lack of affordable, automated solutions for smaller plants
- Absence of real-time safety feedback mechanisms

Methodology Overview:

1. **Problem Definition:** Design a user-friendly, RF-controlled slag remover that eliminates manual interaction with molten metal.
2. **System Design:** Develop mechanical and IoT subsystems for actuation, temperature monitoring, and user feedback.
3. **Component Integration:** Assemble sensors, actuators, and control boards with efficient wiring and coding.
4. **Testing and Evaluation:** Conduct performance evaluations under simulated slag removal conditions.

Tools & Technologies Used:

- Arduino Uno
- IR sensors & remote
- DC motors & L293D motor driver
- TMP36 temperature sensor
- LCD (16x2)
- Visual and sound indicators (LEDs & Piezo)

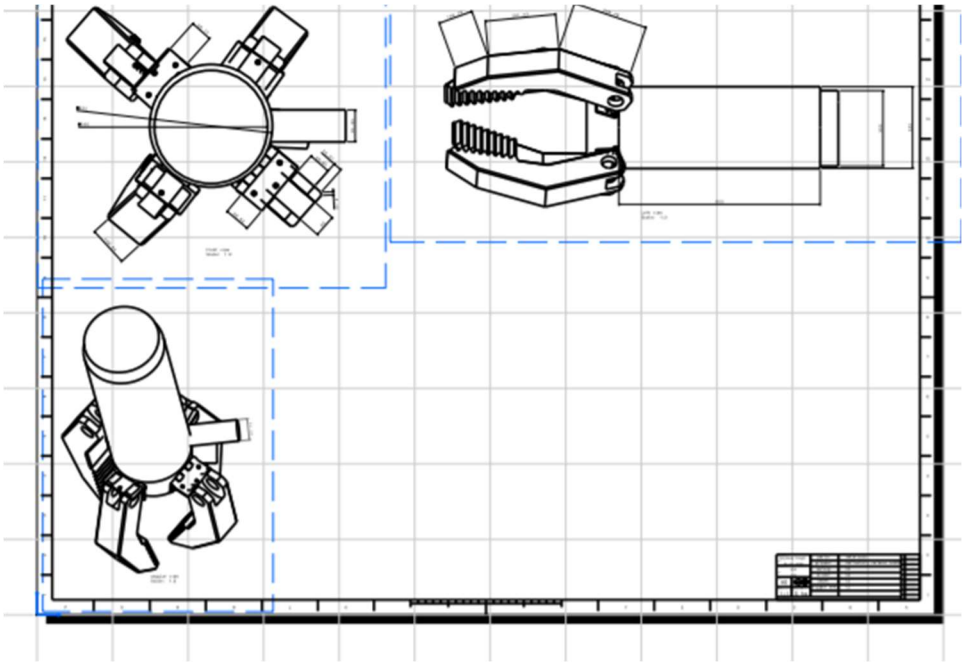
---

3. Proposed Design to Solve the Problem

Mechanical Design

Specifications:

- Material: Mild Steel and Cast Iron
- Coating: Aluminum Oxide (Al2O3) on tongs' tips
- Total Weight: Approximately 7 kg

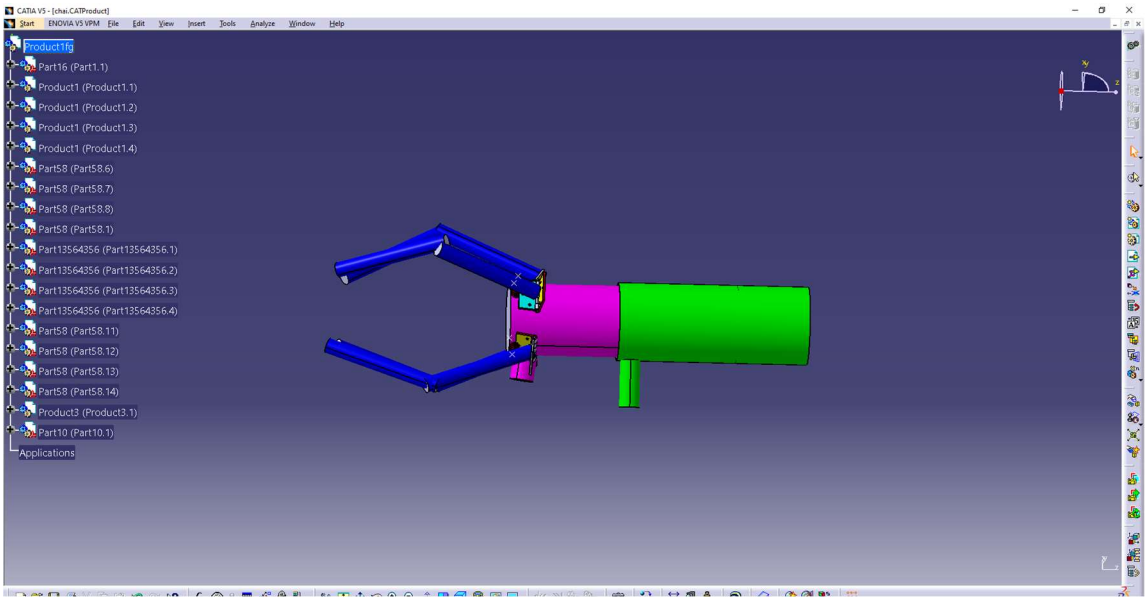


Components:

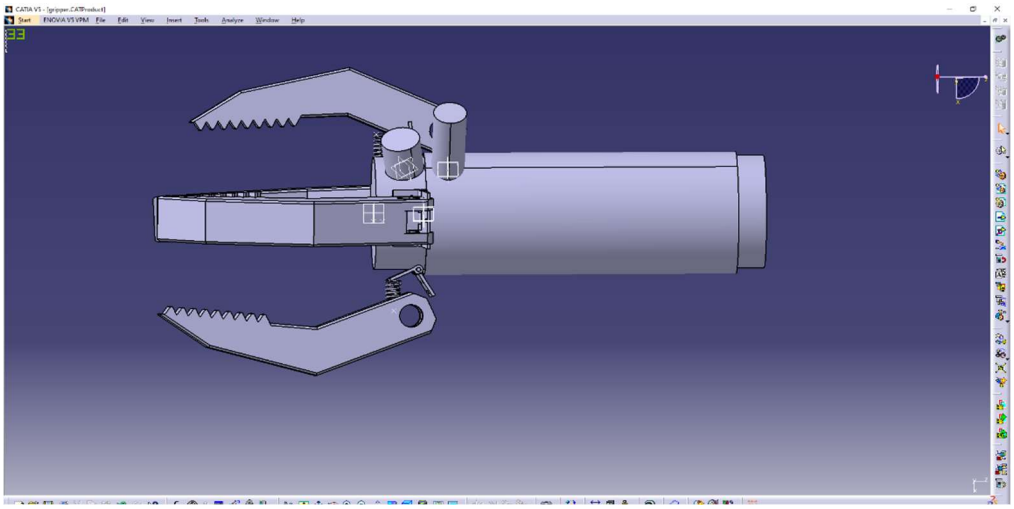
- Hollow cylinder
- Slider mechanism
- Four hinges
- Tongs (Al2O3 coated)
- Four springs

3D mechanical Designs

First design

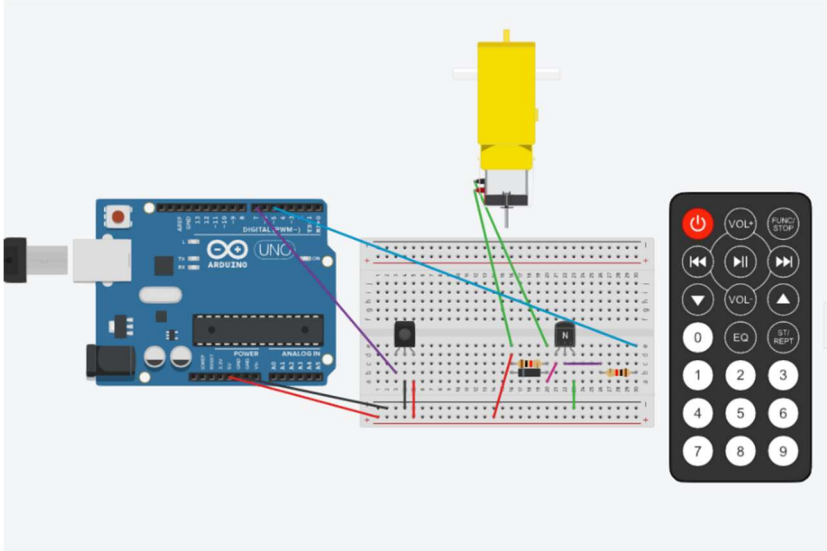


Second design (Better one)



Working: The mechanical system uses a spring-loaded tong assembly mounted on a hollow cylinder. Hinges and sliders are used for motion control. Upon actuation by motor, the tongs close and grab slag safely, avoiding contact with molten metal.

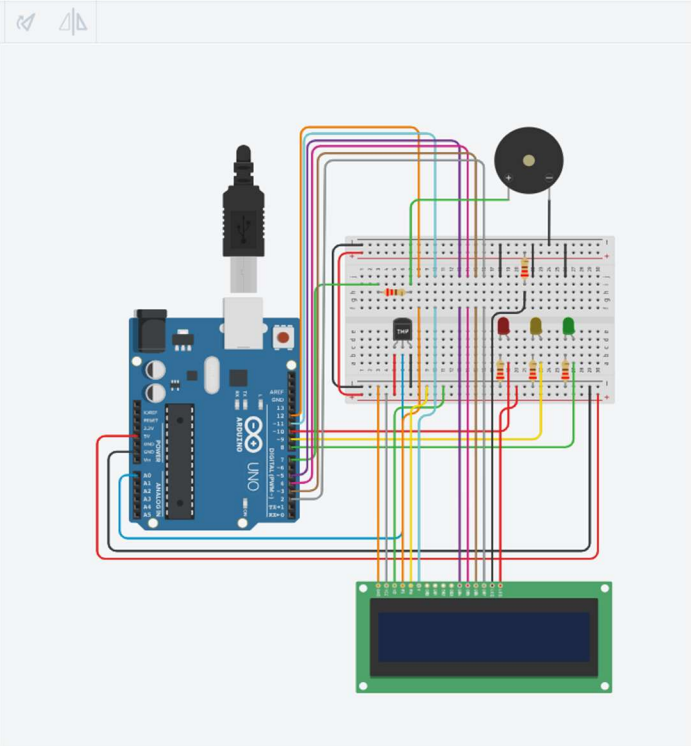
IoT System Design



Components in this design

Name	Quantity	Component
U2	1	Arduino Uno R3
R2 R1	2	1 kΩ Resistor
D2	1	Diode
U3	1	IR sensor
M1	1	Hobby Gearmotor
T1	1	NPN Transistor (BJT)

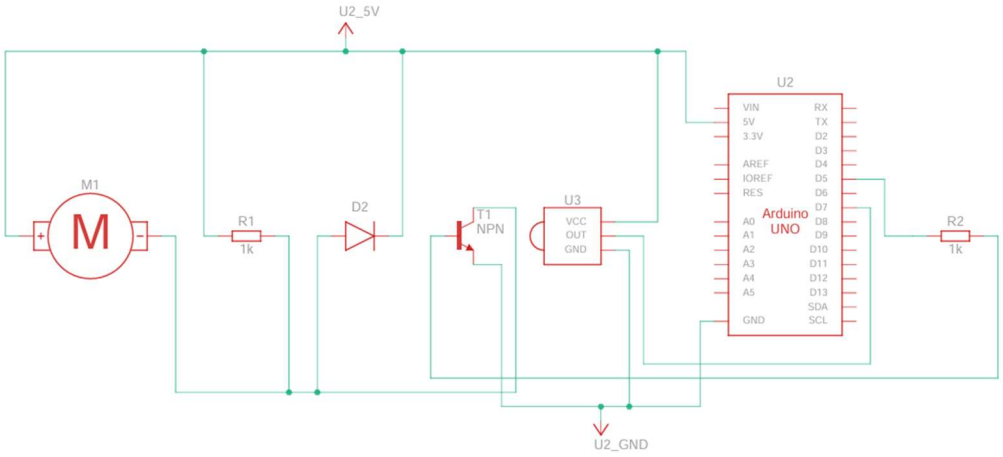
Temperature sensor with Alarm system design



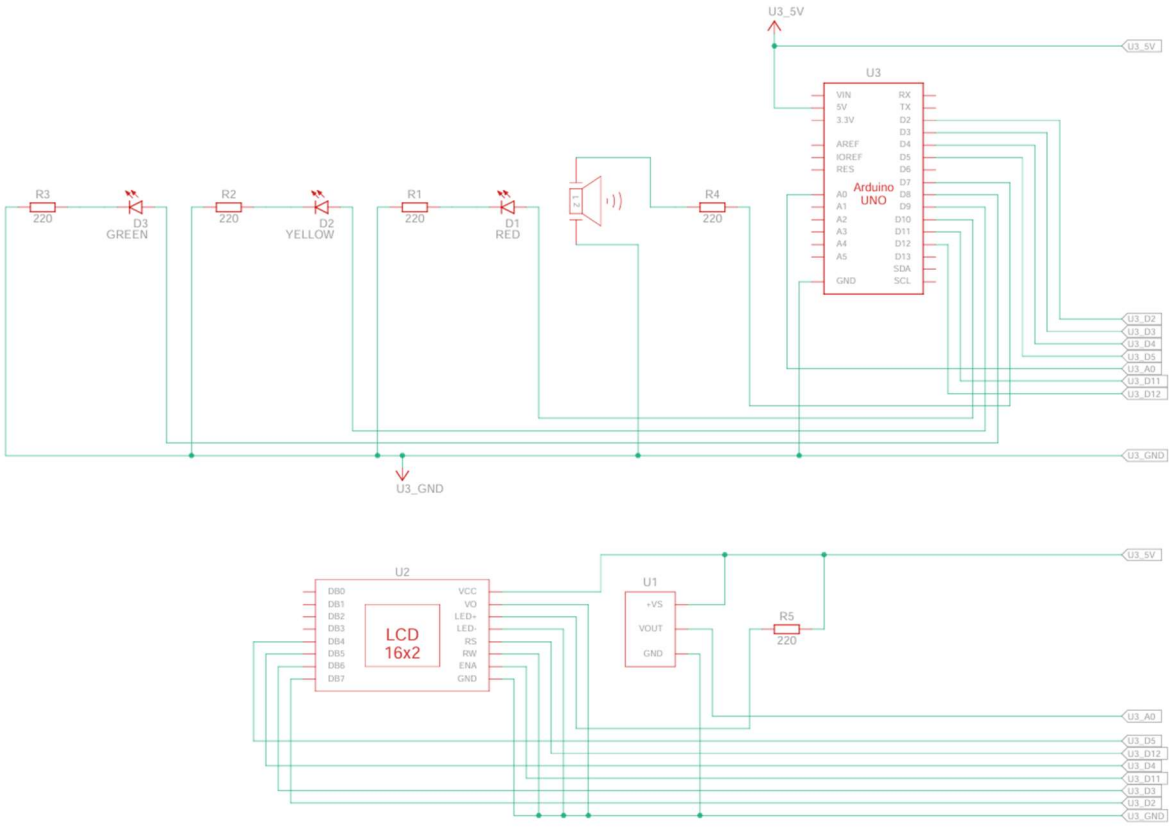
Components in Temperature alarm system

Name	Quantity	Component
U3	1	Arduino Uno R3
U1	1	Temperature Sensor [TMP36]
D1	1	Red LED
D2	1	Yellow LED
D3	1	Green LED
R1 R2 R3 R4 R5	5	220 Ω Resistor
PIEZO1	1	Piezo
U2	1	LCD 16 x 2

Schematic diagram for IoT system



Schematic diagram for temperature alert system



Connections for IoT system

- Connect enable pins (Pin 1, Pin 2) of L293D to 5V output of Arduino. This enables two H-Bridge channels inside the IC to drive two DC motors.
- Connect logic voltage input (Pin 16) of L923D to 5V output of Arduino. This defines the voltage (5V) logic of control signals .
- Connect motor/drive supply (Pin 8) of L293D to +ive of the 9V battery.
- Connect ground pins (Pin 4, 5, 12, 13) to ground of Arduino and -ive of the battery.
- Connect pin 2 of L293D to digital pin 6 of the Arduino.
- Connect pin 7 of L293D to digital pin 5 of the Arduino.
- Connect pin 10 of L293D to digital pin 11 of Arduino.
- Connect pin 15 of L293D to digital pin 12 of Arduino
- Connect first DC motor to Pin 3 and Pin 6 of L293D.
- Connect second DC motor to Pin 11 and Pin 14 of L293D.

Wiring Temperature Alarm System

- Connect **VOUT** of **LM35 sensor** to **Analog pin A0** on Arduino Uno.
- Connect **+VS** of LM35 to **5V** on Arduino.
- Connect **GND** of LM35 to **GND** on Arduino.
- Connect **Red LED** anode to **Digital pin D2** via **220Ω resistor (R1)**. Cathode to GND.
- Connect **Yellow LED** anode to **Digital pin D3** via **220Ω resistor (R2)**. Cathode to GND.
- Connect **Green LED** anode to **Digital pin D4** via **220Ω resistor (R3)**. Cathode to GND.
- Connect **RS** of LCD to **Digital pin D5** on Arduino.
- Connect **EN (Enable)** of LCD to **Digital pin D6**.
- Connect **DB4 (D4)** of LCD to **Digital pin D7**.
- Connect **DB5 (D5)** of LCD to **Digital pin D8**.
- Connect **DB6 (D6)** of LCD to **Digital pin D9**.
- Connect **DB7 (D7)** of LCD to **Digital pin D10**.
- Connect **RW** pin of LCD to **GND** (for write mode).
- Connect **VSS** to **GND**, **VDD** to **5V** on Arduino.
- Connect **VO** (contrast) via **220Ω resistor (R5)** between **GND and 5V**.
- Connect **LED+** of LCD (backlight) to **5V**, and **LED−** to **GND**.

Coding for both systems :

Code for IoT system

```
#include <SoftwareSerial.h>
#include <IRremote.h>
  #include <IRremote.h>

int RECV_PIN =  7;
IRrecv irrecv(RECV_PIN);
decode_results results;
const int  MOTOR=5;   //Motor on Digital Pin 5

void setup()
{
 Serial.begin(9600);
 irrecv.enableIRIn(); // Start the receiver
   Serial.println(results.value, HEX);
   irrecv.resume(); // Receive the next value

 pinMode(MOTOR,OUTPUT);
}

void loop()
{
```

```
 if (irrecv.decode(&results)) {
   Serial.println(results.value, HEX);
   irrecv.resume(); // Receive the next value


 for (int i=0;   i<256;   i++)
{
 analogWrite(MOTOR,i);
 delay(10);
}
delay(2000);
for  (int i=255;  i>=0;  i--)
 {
  analogWrite(MOTOR, i);
  delay(10);
 }
delay(2000);
}
}
```

Code for Temperature alarm sensor

```
#include <LiquidCrystal.h>


// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int tmp = A0;

int baselineTemp = 0;

int celsius = 0;

int fahrenheit = 0;


void setup()

{

lcd.begin(16, 2);

  pinMode(tmp, INPUT);

pinMode(A0, INPUT);

  Serial.begin(9600);


  pinMode(8, OUTPUT);

  pinMode(9, OUTPUT);

  pinMode(10, OUTPUT);

  pinMode(7, OUTPUT);

}
```

```
void loop()
{
baselineTemp = 5;

  celsius = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125);

   fahrenheit = ((celsius * 9) / 5 + 32);
  Serial.print(celsius);
  Serial.print(" C, ");
  Serial.print(fahrenheit);
  Serial.println(" F");

   if (celsius < baselineTemp)
   {
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
     tone(7, 220);

  }
  if (celsius >= baselineTemp && celsius < baselineTemp + 35)
  {
    digitalWrite(8, HIGH);
```

```
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
}

if (celsius >= baselineTemp + 35 && celsius < baselineTemp + 45)

{
  digitalWrite(9, HIGH);
   delay(1000);
  digitalWrite(9, LOW);
  digitalWrite(8, LOW);
  digitalWrite(10, LOW);
  tone(7, 220, 100);
  delay(500);
}

if (celsius >= baselineTemp + 45)
{
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  tone(7, 220);
 }
```

```
   lcd.clear();
int reading = analogRead(tmp);
  float voltage = reading * 5.0;
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100 ;


  if (celsius < baselineTemp)
  {
  lcd.print("DANGER ZONE");
  lcd.setCursor(0,1);
  lcd.print(temperatureC);
  lcd.println(" degrees C ");
  delay(1000);
  }


  if (celsius >= baselineTemp && celsius < baselineTemp + 35)
  {
  lcd.print("SAFE ZONE");
  lcd.setCursor(0,1);
  lcd.print(temperatureC);
  lcd.println(" degrees C ");
```

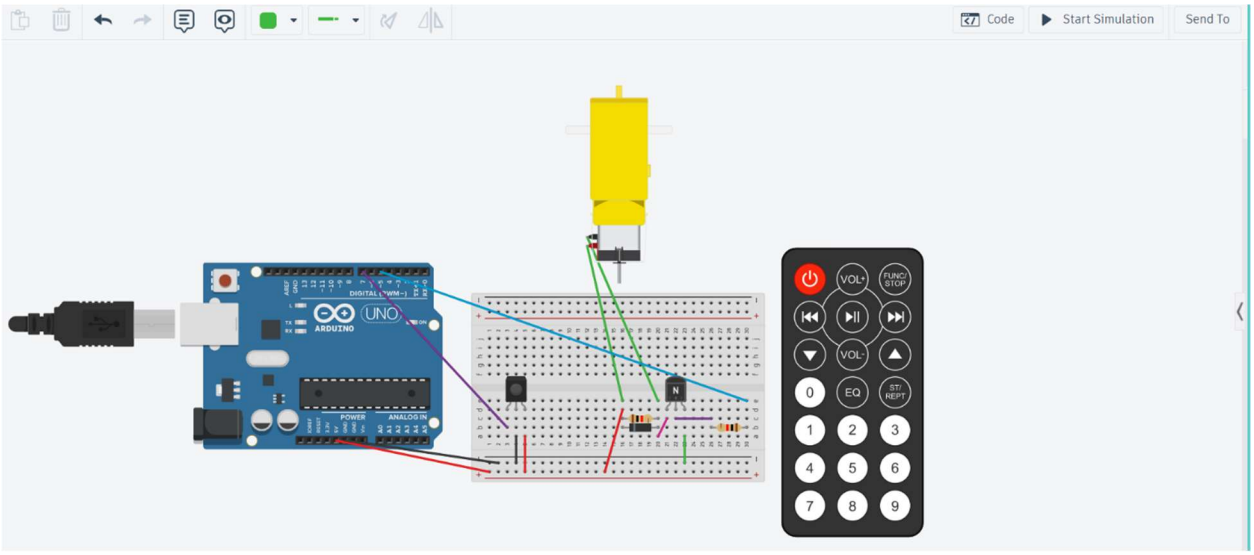
```
  delay(1000);
  }

  if (celsius >= baselineTemp + 35 && celsius < baselineTemp + 45)
  {
  lcd.print("CAUTION ZONE");
  lcd.setCursor(0,1);
  lcd.print(temperatureC);
  lcd.println(" degrees C ");
  delay(1000);
  }

  if (celsius >= baselineTemp + 45)
  {
  lcd.print("DANGER ZONE");
  lcd.setCursor(0,1);
  lcd.print(temperatureC);
  lcd.println(" degrees C ");
  delay(1000);
  }
}
```

Performance Evaluation Results

Hard ware simulation -

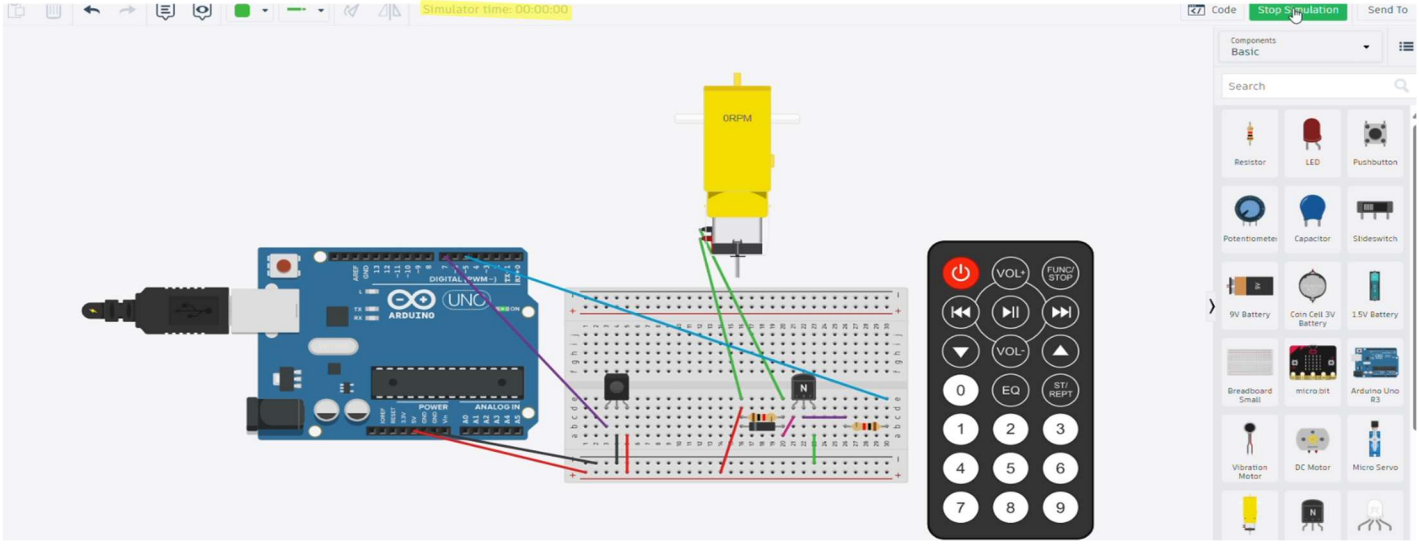
For Iot System –

Initially system at rest



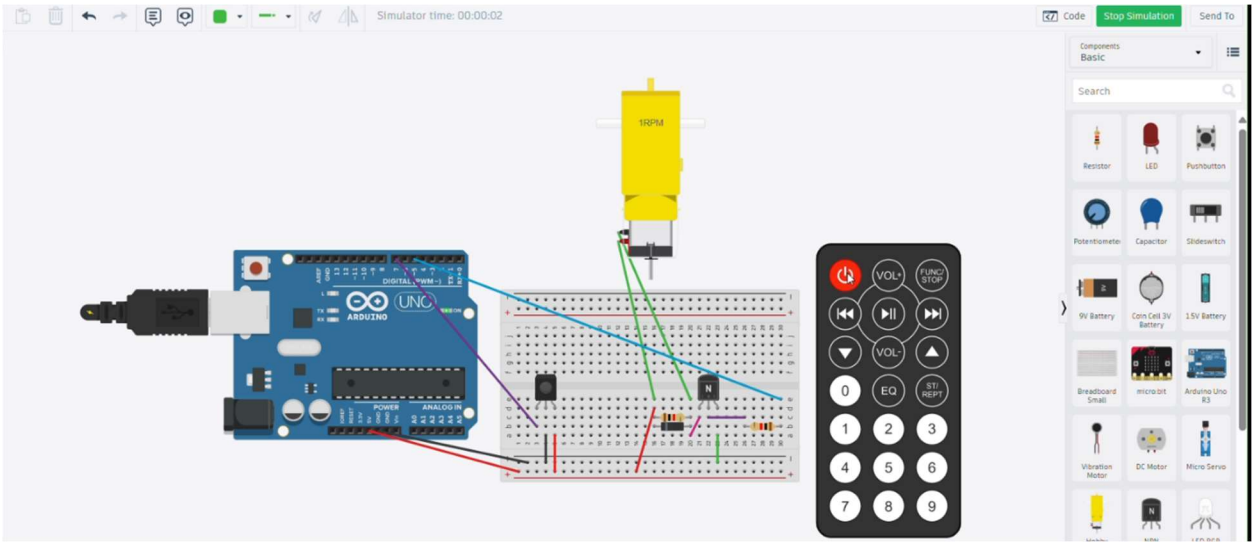
When pressed button on Infra red remote, it sends infra red signal to IR sensor which makes the motor rotates which in turns activates the sliding mechanism on slag removal tool which in turn activates the tongs to act as hand like mechanism which can take out the slag from the pouring ladle.

Now starting the simulation
Simulation time 00:00:00



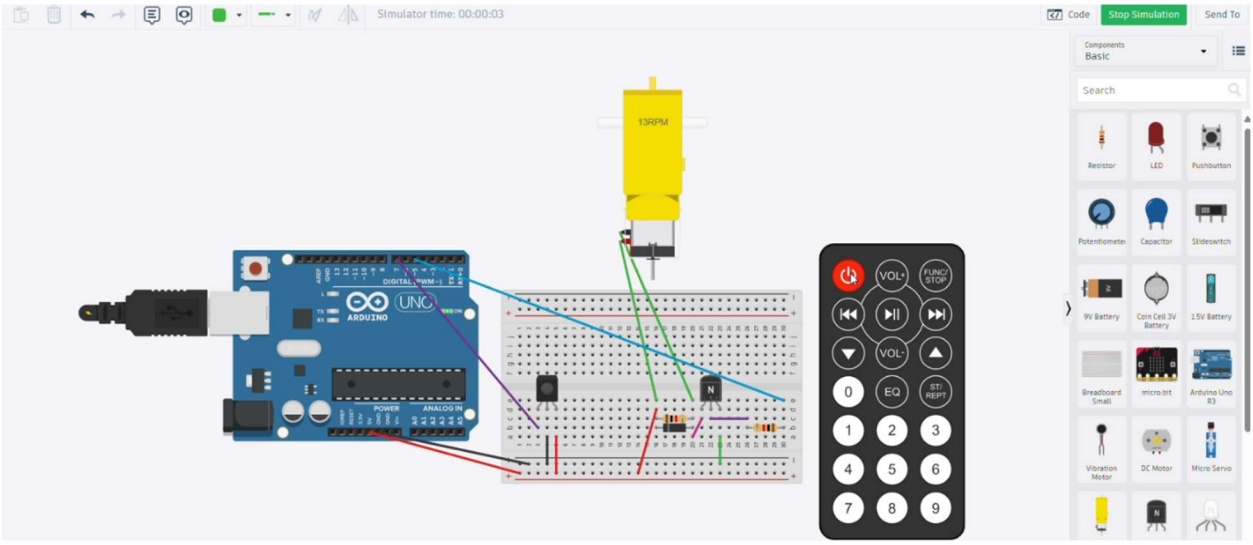
No Effect , Rotor does not rotate , No signal given or received.

Simulation time 00:00:02



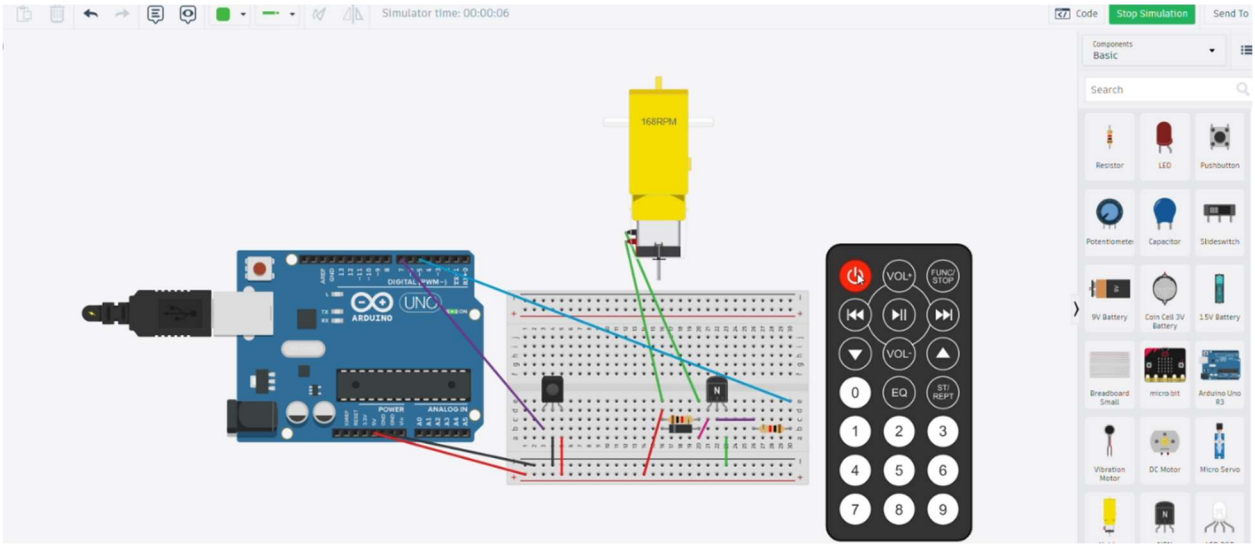
When pressed button on IR (Infra red Remote) , IR sensor receives the signal and Rotor starts rotating with 1 RPM (rotation per minute) that means Signal is received by IR sensor and system is working.

Simulation time 00:00:03



Rotor has started to pickup the speed and it is working properly.
Now we are going to see simulation at time 00:00:06

Simulation at 00:00:06



Rotor has achieved maximum rpm 168, and at this rpm slag removal can be activated through motor control system and it can achieve it's sliding mechanism on hollow cylinder and thus which activates hand like mechanism at tongs due to hinges and spring.

At this point, mechanical design will start working, it can use hand like structure of tongs and using that it can take impurity from pouring ladle and throw in garbage.

This will increase industrial safety as temperature at Foundry plants are huge, by removing the manual process of skimming the slag from ladle it can protect lives and save from hazards.
Lot of accidents occur in Foundry plants mainly at melting shops were metal is melted which is to be used to make mold and final product of any mechanical component.

By removing the manual process and making workers work from afar from the hazardous environment, this tool can develop industrial safety at highest standards and save life.

By efficiently removing slag it can remove slag/ impurities which can make products better with less defects. Defects in any mechanical component can occur during manufacturing process and these arise from many reasons like pattern tolerances not accurate or molten metal temperature not up to design standard but primarily they arises from impurity or slag. By removing slag , it makes the product defect free which in turn can increase product life cycle as defects can deteriorate life span of a product. It can make quality control and testing easy as less defects, less in depth checking for products and less labour which is very beneficial for industry.

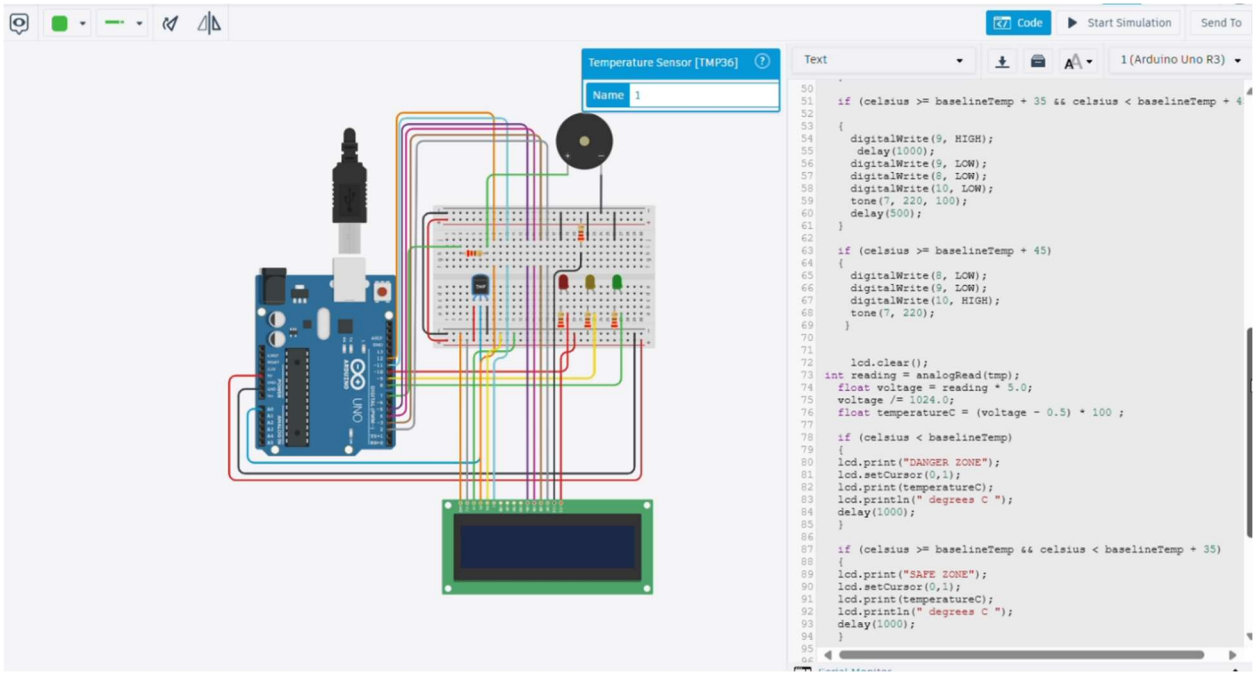
Operations of the slag removal tool is environmental friendly as it does not produces any GHG emissions and there are no carbon foot prints. Thus by having compatibility with socio factors like increasing safety for workers , by benefiting the industry in economical factors and environmental factor , this tool can be deemed sustainable.

See mechanical design and working for more clarity.

Similarly, it can be designed for rotor to rotate in opposite direction and it will come to its original position after throwing the slag into garbage.

Hardware simulation for Temperature alert system :

————When system at rest :



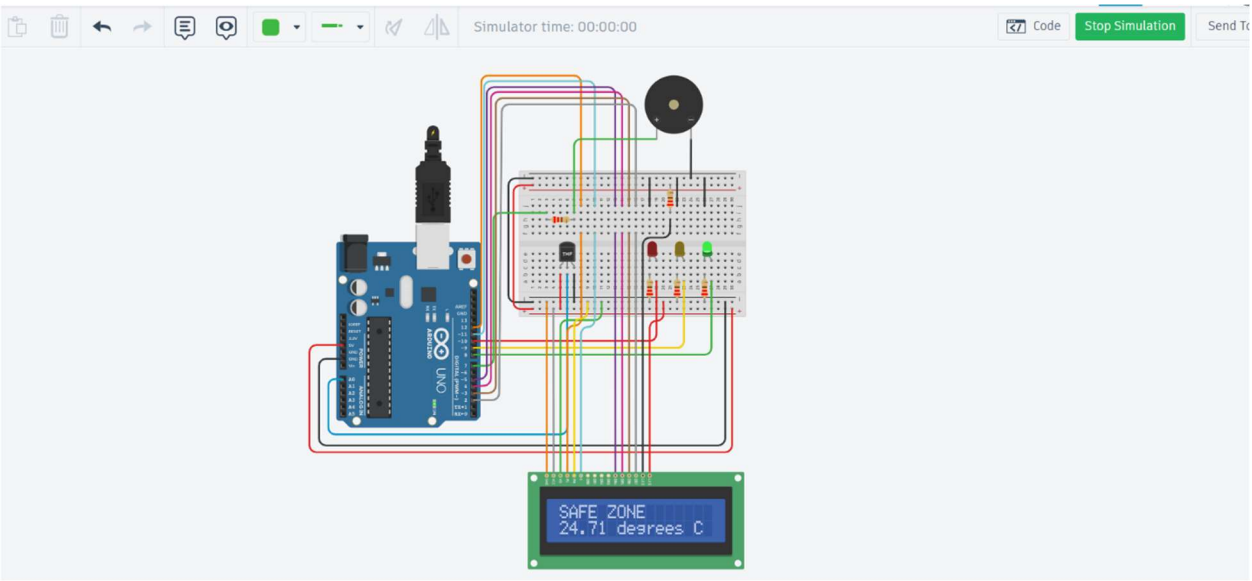
System has not been activated and is at rest , it will be activated when we initiate the IR sensor from previous design, when IoT system starts and tongs are working, temperature alert system starts working to safe guard the tool by indicating the temperature on the tool through LCD and a sound alarm system.

For the initial simulations I have set the temperature ranges as:

Zone Name	Temperature Range
DANGER ZONE	Below **5°C**
SAFE ZONE	**5°C** to **39°C**
CAUTION ZONE	**40°C** to **49°C**
DANGER ZONE	**50°C and above**

The LCD displays zone names and live temperature (converted from voltage). In the real setting , temperature will be at different range and for that different sensor have to be used with advanced shielding from higher temperatures. This is alarm system can serve as foundation ground for the advanced alarm system for higher temperature in foundry.

Simulation at time 00:00:00 , For safe zone -

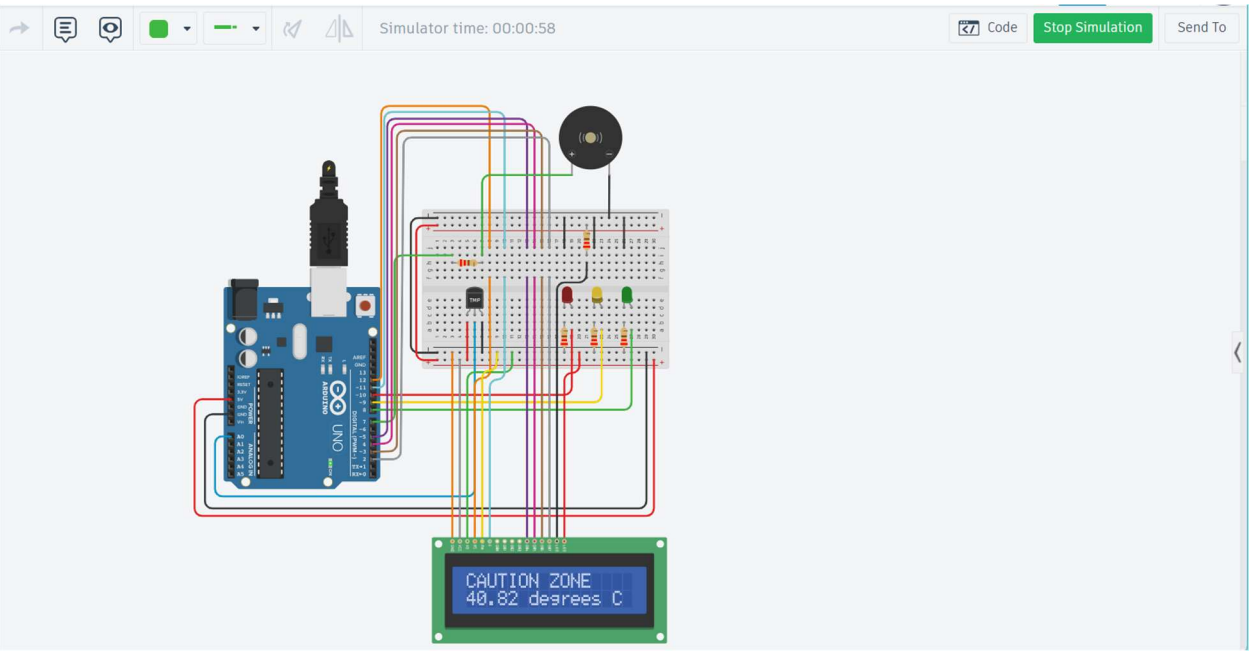


When the previous IoT system gives IR signal to actuates the motor for slag removal tool to work, the temperature alert system turns on and at room temperature , meaning at time 0 it is indicating 24.71 degrees Celsius which is safe zone. To indicate if it's a safe zone green led is lit and on LCD it displays Safe zone.

Now when slag removal tool has started working and it has picked up the impurities from the pouring ladle, it's temperature will generally go higher because of the working conditions.

To safe guard the tool from thermal damage, I have set up the caution zone in between 40 degrees Celsius to 49 degrees Celsius.

Simulation at time 00:00:58 for caution zone at 40.82 degrees celcius-

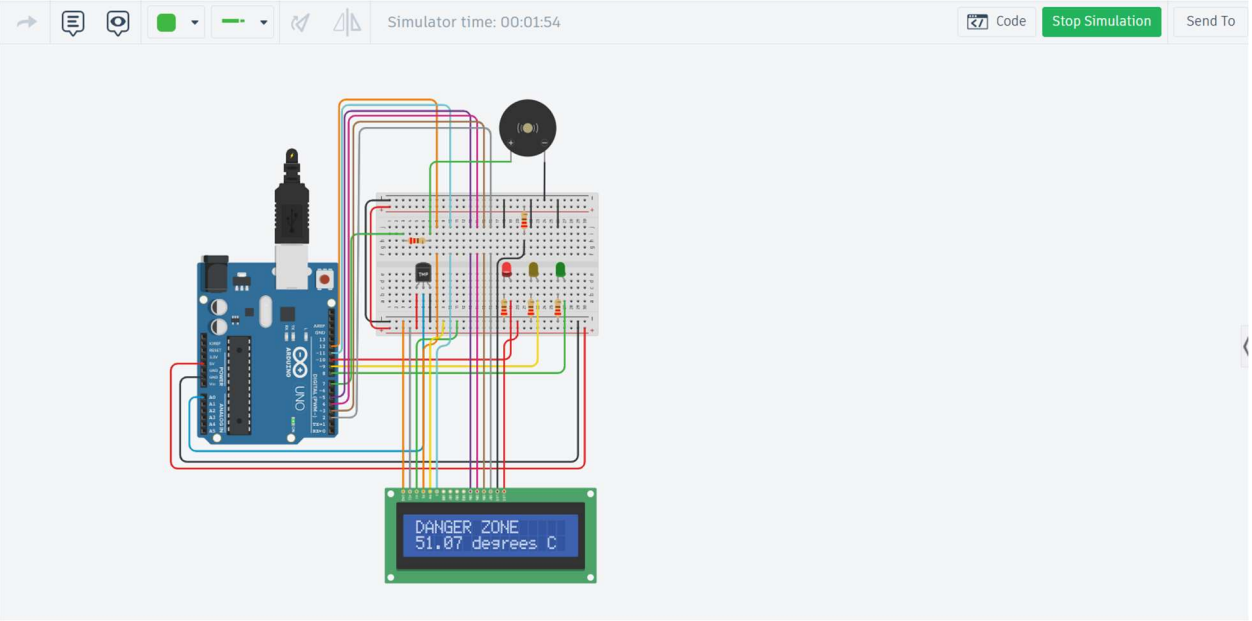


Now that Slag removal tool has been used for while and it has build up some thermal residue, for the safeguard of the tool there should be tolerance limit which the tool can handle, and I have set to 40 degrees Celsius to 49 degrees Celsius. In caution zone, the system starts to make beep sound at burst and it displays yellow led signal with LCD display as " Caution Zone".

By caution zone, we can have a cool down for tool to let it remove thermal residue so it can go back to safe zone and then we can use the tool. This serve as a safety barrier and great method for industrial safety.

Suppose the tool had been exerted to extreme temperature which are hazardous and can damage the tool. For that I have designed danger zone starting from 50 degree Celsius or above. Again for real settings, this will be a different temperature range, this only serve as foundation grounds for the initial designs.

Simulation time 00:01:54 for Danger zone at 51.07 degree Celsius.



Here in this simulation, the tool has been exerted to dangerous thermal residue which is 51.07 degree Celsius and at this temperature it can seriously harm the equipment and to safeguard the equipment from damage, it will alert the operator to stop the process immediately by displaying “DANGER ZONE” at LCD system with a red led. Additionally it will make a very sharp beep sound through “PIEZO1” which is a loud speaker. It can let operator know that the equipment has reached extreme temperature range.

This kind of alarm system serves as industrial safety mechanism which is essential for safety of mechanical equipment. Industries can save thousands of dollars by implementing industrial safety mechanisms like these so they don’t have to spend more on repair, maintain and buying the new equipment for industrial process.

Testing Methodology:

Temperature sensor in Simulink
Simulating 3 simulations for the LM35-based Temperature Alarm System in Simulink, validated logic using **Data Inspector plots** simulation screenshots

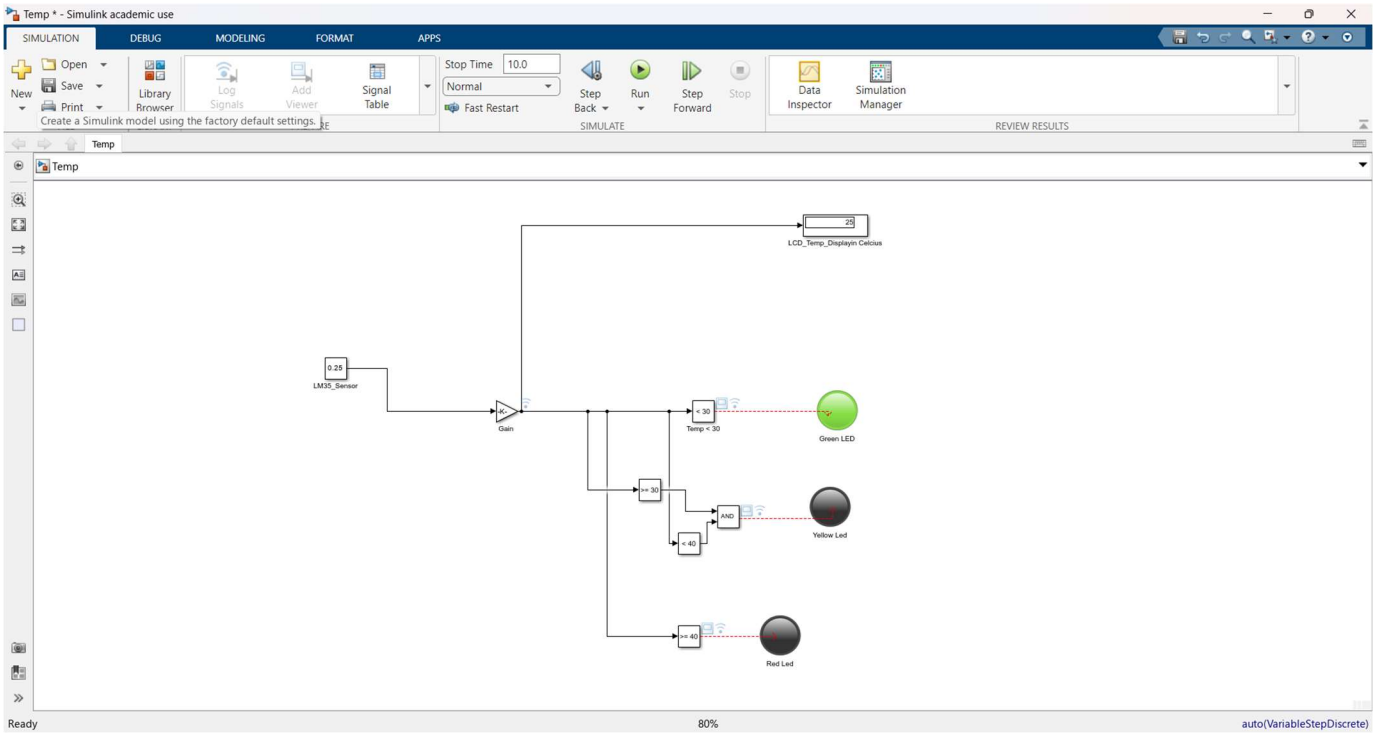
Simulation 1

LM35 = 0.25V (→ 25°C)

Expected Behavior:

- 25°C < 30°C → **Green LED should be ON**
- Yellow and Red LEDs should remain OFF
- LCD should show 25

Simulation screenshot from Simulink



Simulation 1 results -

- Green LED Activation:
- The LM35 sensor block was set to output 0.25V, which when passed through the Gain block (×100), resulted in an output of 25°C.
- According to the system logic, temperatures below 30°C should activate the Green LED.
- In the Simulink simulation window, the Green LED indicator is lit, confirming that the < 30°C condition was successfully triggered.
- Yellow and Red LEDs Remain OFF:
- The Yellow LED is governed by a logical AND condition, where the temperature must be greater than or equal to 30°C and less than 40°C.
- Since 25°C is less than 30°C, this AND condition evaluates to false, hence the Yellow LED remains off.
- Similarly, the Red LED, controlled by the condition Temp ≥ 40°C, remains off, as 25°C does not satisfy this threshold.
- LCD Display Verification:
- The temperature value after amplification is routed to the LCD display block, which is configured to display the current temperature in Celsius.
- The simulation shows “25” on the LCD, confirming that:
- The analog voltage-to-temperature conversion was accurate.
- The data path between the sensor and display is functioning as intended.
- Simulation Data Inspector Insights:
- The Data Inspector was used to monitor key logical signals during simulation runtime.
- The signal representing the condition Temp < 30 maintained a steady value of 1 (True) throughout the 10-second simulation, confirming the trigger.
- All other logical condition signals:
- Temp ≥ 30 evaluated to 1 (True), consistent with the input condition.
- Temp < 40 evaluated to 0 (False), consistent with the input condition.
- Temp ≥ 40 evaluated to 0 (False), consistent with the input condition.
-

Conclusion:

- This simulation successfully validates the lower-bound temperature alert functionality of the system.
- The Simulink comparison blocks are behaving exactly as expected:
- Proper analog-to-digital conversion through the Gain block.
- Precise conditional branching using logic comparators.
- The LCD module is correctly interpreting the input and displaying the accurate temperature value.
- The Data Inspector serves as an essential debugging and validation tool, providing confidence that internal signal logic aligns with external behavior.
- This scenario demonstrates that the system is highly responsive and accurate when monitoring temperatures below the first threshold (30°C) — which is critical in any temperature monitoring or safety system.

The following signals were monitored during simulation:

Signal Name	Color	Description
Compare To Constant2:1	Red	Represents Temp >= 40 (Red LED logic)
Gain:1	Blue	Output of Gain block = Scaled temperature (°C)
Logical Operator:1	Yellow	AND logic for Yellow LED (Temp >= 30 AND Temp < 40)
Temp < 30:1	Purple	Comparator for Green LED activation

Data Inspector Results from Simulation 1
Interpreting the chart:

1. Blue Line (Gain Output – 25°C):
   - Constant throughout the simulation (from t = 0 to t = 10s).
   - Confirms that the sensor voltage was steady and translated to 25°C correctly.
2. Purple Line (Temp < 30) – Value: 1
   - This line stays high (1), meaning the < 30°C condition is True for the entire simulation.
   - Result: Green LED is ON → Working correctly
3. Red Line (Temp >= 40) – Value: 0

    - Stays low (0), showing this condition was not triggered.
    - Result: Red LED remains OFF → Correct
4. Yellow Line (AND Logic for Yellow LED) – Value: 0
    - Also remains at 0, since Temp >= 30 was False and Temp < 40 was True, but the AND logic failed.
    - Result: Yellow LED remains OFF → Valid

Validation Summary

Condition	Status	Output Logic	Result
Temp = 25°C	True	From Gain	Accurate conversion
Temp < 30	True	1 (True)	Green LED ON
Temp ≥ 30 AND < 40	False	0 (False)	Yellow LED OFF
Temp ≥ 40	False	0 (False)	Red LED OFF

Conclusion

The Data Inspector confirms that:

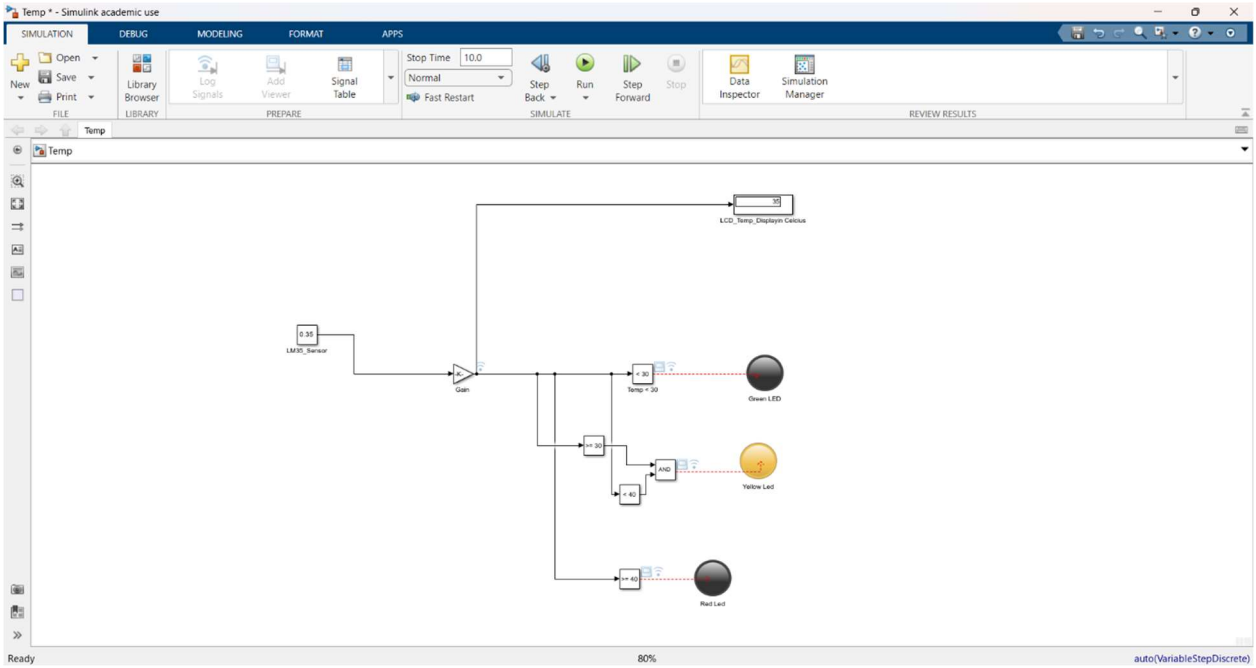
- Only the Green LED logic is triggered.
- Other conditions are inactive, as they should be.

Simulation 2: LM35 = 0.35V (→ 35°C)

Expected Behavior:

- 30°C ≤ 35°C < 40°C → Yellow LED should be ON
- Green and Red LEDs should remain OFF
- LCD should show 35

Simulation 2 in Simulink



Simulation 2 Results – LM35 Input: 0.35V (→ 35°C)

Yellow LED Activation:

- The LM35 sensor block was set to output 0.35V, which, when passed through the Gain block (×100), resulted in an output of 35°C.
- According to the system logic, the Yellow LED should activate when the temperature is greater than or equal to 30°C and less than 40°C.
- In the Simulink simulation window, the Yellow LED indicator is lit, confirming that the condition 30°C ≤ Temp < 40°C was successfully triggered.

Green and Red LEDs Remain OFF:

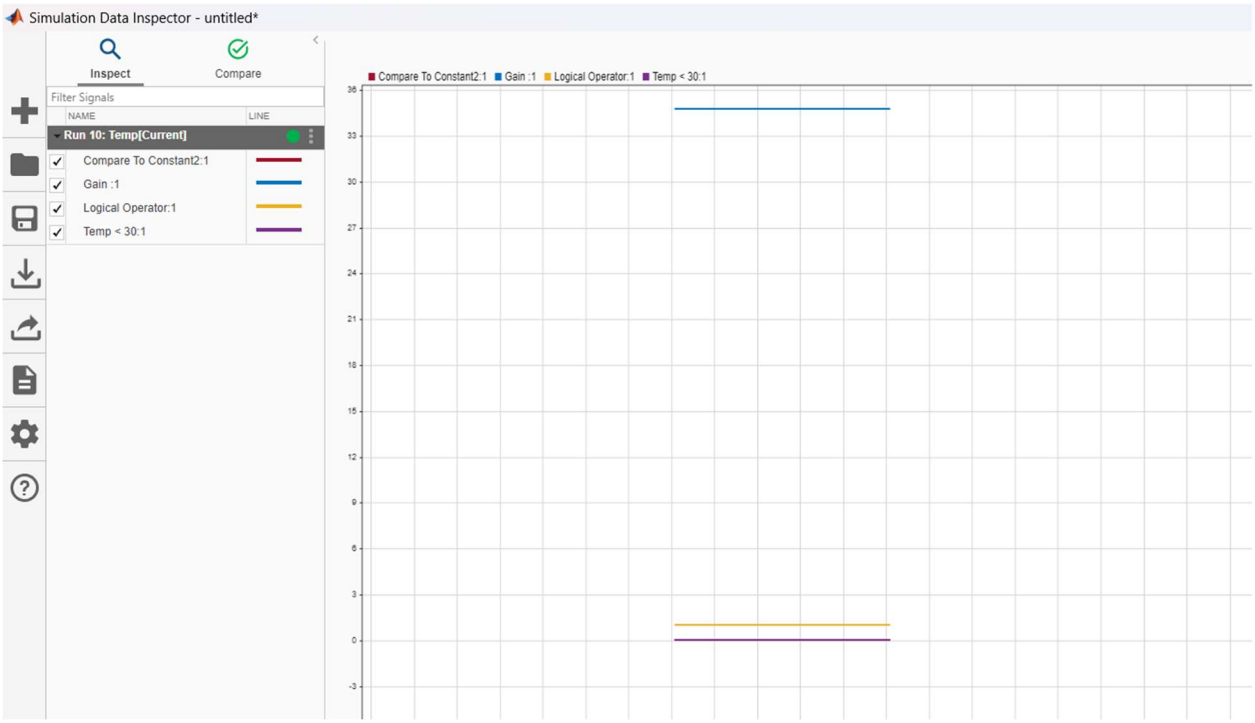
- The Green LED is governed by the condition Temp < 30°C. Since 35°C > 30°C, this condition evaluates to false, keeping the Green LED off.

- The Red LED is governed by the condition Temp ≥ 40°C. Since 35°C < 40°C, this condition also evaluates to false, so the Red LED remains off.
- Therefore, only the Yellow LED responds to this mid-range temperature, as intended.

LCD Display Verification:

- The temperature value after amplification is routed to the LCD display block, which is configured to show the current temperature in Celsius.
- The simulation shows "35" on the LCD, confirming that:
- The voltage-to-temperature conversion is functioning correctly.
- The signal pathway from sensor input to visual display is working without interruption.

Data Inspector results-



Logged Signals Overview

Signal Name	Color	Description
Compare To Constant2:1	Red	Temp ≥ 40°C → Red LED trigger
Gain:1	Blue	Output temperature in Celsius after amplification
Logical Operator:1	Yellow	AND condition for Yellow LED: Temp ≥ 30 AND Temp < 40
Temp < 30:1	Purple	Green LED condition

Graph Interpretation

1. Blue Line (Gain Output = 35):

- Constant throughout the simulation duration.
- Confirms that **0.35V was accurately converted to 35°C** using the Gain block (×100).
- This temperature value is used across all logic blocks and the LCD display.

2. Yellow Line (Logical Operator for Yellow LED):

- Stays at **1 (True)** from t = 0 to t = 10 seconds.
- Indicates that the condition **Temp ≥ 30 AND Temp < 40** was satisfied.
- Confirms the **Yellow LED was correctly triggered**.

3. Purple Line (Temp < 30):

- Stays at **0 (False)** throughout the simulation.
- This is expected, as the temperature is **35°C**, not less than 30°C.
- Confirms that the **Green LED condition was not met**, and it correctly remained OFF.

4. Red Line (Temp ≥ 40) – Compare to Constant2:1:

- Also at **0 (False)**.
- Since **35°C < 40°C**, this is correct.
- Confirms that the **Red LED did not trigger**, which matches expected logic.

Summary of Signal Logic

Condition	Signal Value	LED Response
Temp = 35°C	35 (Blue Line)	35
Temp < 30	0 (Purple)	Green LED OFF
Temp ≥ 30 AND Temp < 40	1 (Yellow)	Yellow LED ON
Temp ≥ 40	0 (Red)	Red LED OFF

Conclusion

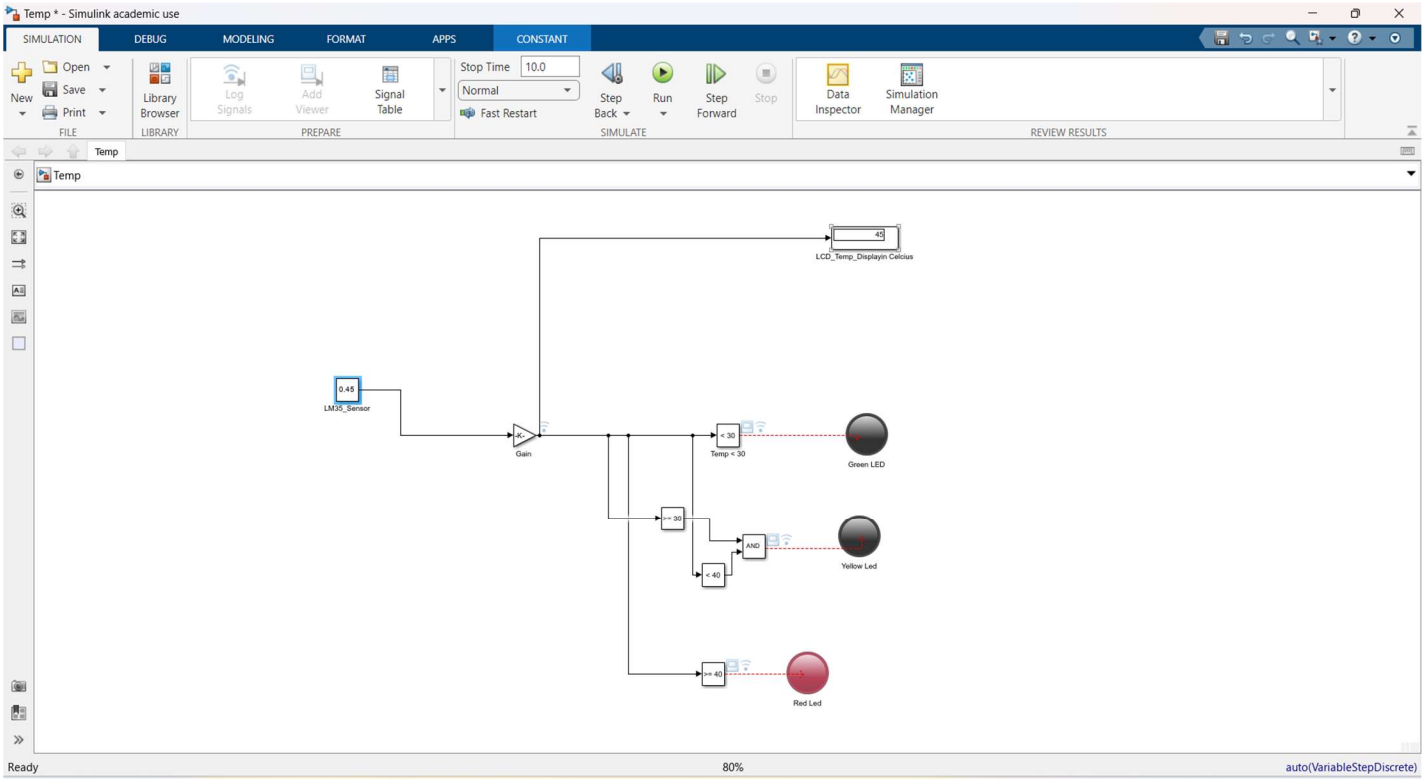
- The **Data Inspector results strongly validate** that the **Yellow LED** logic was activated **only** when both conditions Temp ≥ 30 and Temp < 40 were satisfied.
- The **Green and Red LEDs remained off**, as expected.
- This reinforces that the logical AND operation is functioning with **precision** and the system is handling **mid-temperature range (warning state)** exactly as intended.
- The **Gain block output is stable**, ensuring consistent behavior throughout the simulation period.

Simulation 3: LM35 = 0.45V (→ 45°C)

Expected Behavior:

- 45°C ≥ 40°C → Red LED should be ON
- Green and Yellow LEDs should remain OFF
- LCD should show 45

Simulation screen shot from Simulink



Simulation 3 Results – LM35 Input: 0.45V (→ 45°C)

Red LED Activation:

- The **LM35 sensor block** was configured to output **0.45V**, which was passed through a **Gain block (×100)**, resulting in an output of **45°C**.
- According to the system logic, temperatures **greater than or equal to 40°C** should activate the **Red LED** to indicate a **high-temperature alert**.
- In the **Simulink simulation**, the **Red LED indicator is lit**, confirming that the Temp ≥ 40°C condition was successfully triggered.

Green and Yellow LEDs Remain OFF:

- The **Green LED** is triggered by the condition Temp < 30°C. Since **45°C is above 30°C**, this condition evaluates to **false**, so the Green LED remains **off**.
- The **Yellow LED** is controlled by an **AND condition**:
  Temp ≥ 30°C **AND** Temp < 40°C.
  - While the first part (Temp ≥ 30) is **true**, the second part (Temp < 40) is **false**, resulting in the overall condition being **false**.
  - Therefore, the **Yellow LED also remains off**.

LCD Display Verification:

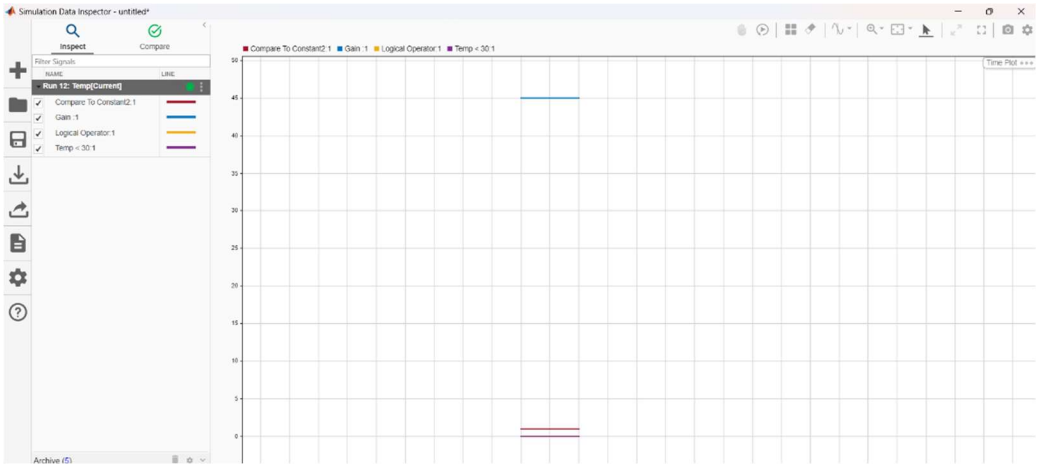
- The temperature value (45°C) is fed into the **LCD display block**, which is configured to show the live temperature in Celsius.
- The LCD displays **"45"**, confirming that:
- The **analog voltage-to-temperature conversion** (0.45V → 45°C) is accurate.
- The **sensor-to-display data flow** is working as intended.

Conclusion:

- This simulation confirms that the system **accurately detects high-temperature conditions** and triggers the **Red LED alert** only when the input temperature reaches or exceeds **40°C**.
- All logical paths behaved as expected:
- Red LED ON
- Green and Yellow LEDs OFF
- The **LCD display accurately reflected the input**, showing a clear temperature value of **45°C**.

  This scenario demonstrates that the system is **well-equipped for high-temperature monitoring**, making it suitable for critical applications like **thermal protection, industrial process alerts, or fire safety systems**.

Simulation Data Inspector -



Simulation 3 – Data Inspector Interpretation

Input Voltage: 0.45V → 45°C

Logged Signals Summary

Signal Name	Color	Description
Compare To Constant2:1	Red	Temp ≥ 40°C → Red LED condition
Gain:1	Blue	Output temperature after Gain block (°C)
Logical Operator:1	Yellow	AND logic for Yellow LED (Temp ≥ 30 AND Temp < 40)
Temp < 30:1	Purple	Condition for Green LED activation

Graph Interpretation

1. Blue Line – Gain Output = 45°C:

- Constant at 45 throughout the simulation.
- Confirms that 0.45V was successfully converted to 45°C via the Gain block (×100).
- This value is used for all conditional logic and display purposes.

2. Red Line – Temp ≥ 40 = 1 (True):

- This signal stays high (1) for the entire duration.
- Confirms that the Red LED condition was satisfied, which is consistent with the 45°C input.

3. Yellow Line – AND Logic for Yellow LED = 0 (False):

- The AND condition (Temp ≥ 30 AND Temp < 40) is False because while Temp ≥ 30 is true, Temp < 40 is False.
- Hence, this signal stays low (0) → Yellow LED OFF

4. Purple Line – Temp < 30 = 0 (False):

- As expected, 45°C > 30°C, so this comparator returns False.
- Confirms Green LED OFF.

Logical Condition Summary

Condition	Signal Value	LED Status
Temp = 45°C (from Gain)	45	—
Temp < 30	0 (False)	Green LED OFF
Temp ≥ 30 AND Temp < 40	0 (False)	Yellow LED OFF
Temp ≥ 40	1 (True)	Red LED ON

Conclusion

The Data Inspector confirms that:

- The Red LED logic (Temp ≥ 40) was correctly triggered and held throughout the simulation.
- The AND logic for Yellow LED failed, as expected, because 45°C is not below 40°C.
- The Green LED logic is correctly false.
- The Gain block output was stable and reliable across the simulation time frame.

This validation shows your system handles high-temperature detection (> 40°C) precisely, activating only the Red LED, which is essential for applications like overheating alerts, fire detection, and equipment shutdown triggers.

5. Conclusion

The automated slag removal tool developed in this project successfully demonstrates the potential for increased safety, efficiency, and automation in foundry operations. By eliminating the need for manual slag removal and integrating real-time temperature monitoring, the system minimizes human risk and enhances operational control.

Future Work:

- Add wireless camera for live visual monitoring
- Implement wireless data logging
- Use industrial-grade temperature sensors for higher accuracy