

**MICROCONTROLLERS FOR MECHATRONICS – MECA442**

Experiment 8: Analog to digital conversion in microcontrollers

**Jana Kabrit1, Oussama Al Ahmad Matar1**

**1** Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

*Abstract*

*The aim of this experiment is to design and simulate a circuit that will display the temperature measured by a temperature sensor on an LCD screen with a fan that turns on after the temperature exceeds a certain threshold. The computation of the temperature to be displayed – based on the input of the TMP36 sensor – will be done using an Arduino Uno. The experiment was successful, and the desired results were obtained.*

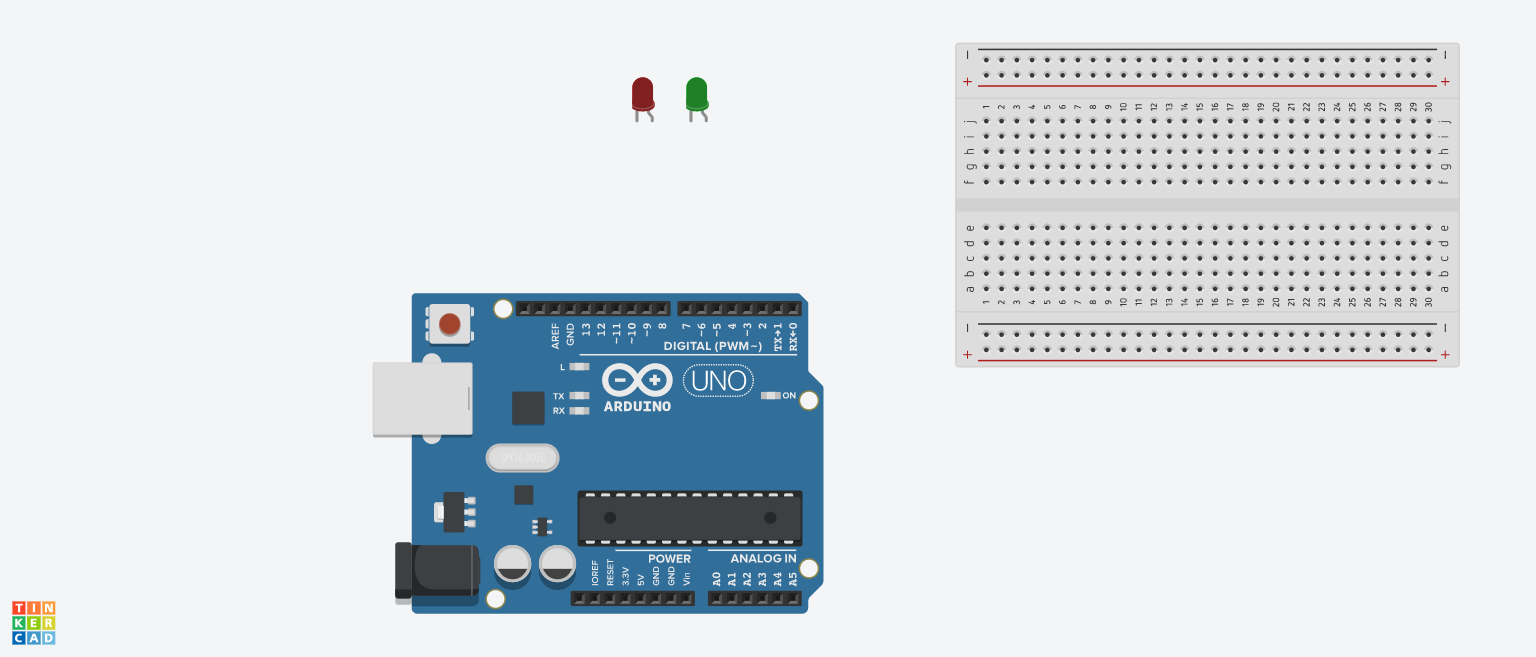
**Keywords:** Arduino Uno, TMP36, Temperature, LCD Screen, TinkerCAD.

1. INTRODUCTION

In this experiment, the temperature read by a TMP36 temperature sensor will be displayed on an LCD screen. The TMP36 reads the temperature of its surrounding and sends it as an analog signal to the Arduino. Thus, the analog signal needs to be converted into a digital one to print the temperature on the screen. To do so, we would read the resolution of the analog signal, and convert it to volts in which every 10mV would represent 1 degree Celsius. The temperature would be displayed on the LCD screen that is connected to the Arduino directly through a breadboard. Once the temperature reading exceeds the threshold of 30° Celsius, a fan will turn on, otherwise, it would remain off. The fan in this experiment was replaced by a buzzer due to simulation limitations.

1. **MATERIALS AND METHODS**
   1. **Materials**
      1. **Simulated Electronics Components**
2. Arduino Uno

The Arduino Uno shown in Figure 1 is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.[1]



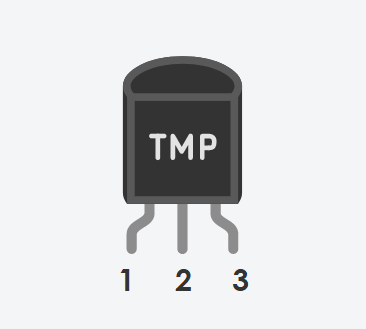
**FIGURE 1:** ARDUINO UNO BOARD

In this experiment, the Arduino Uno is needed to receive input from the temperature sensor, calculate the correct temperature, and display the temperature accurately on a screen.

1. TMP36 Temperature Sensor:

The TMP36 is a low voltage, precision centigrade temperature sensor. It provides a voltage output that is linearly proportional to the Celsius temperature. The TMP36 does not require any external calibration to provide typical accuracies of ±1°C at +25°C and ±2°C over the −40°C to +125°C temperature range. The TMP36 is intended for single-supply operation from 2.7V to 5.5V maximum. It outputs 10mV per degree Celsius, however, requires some software calibration to print the correct temperature.[2]

**FIGURE 2:** TMP36 SENSOR



The sensor has 3 pins as shown in figure 2, a power pin, a VOUT pin for the output voltage proportional to the temperature, and a ground pin.

1. LCD Screen:

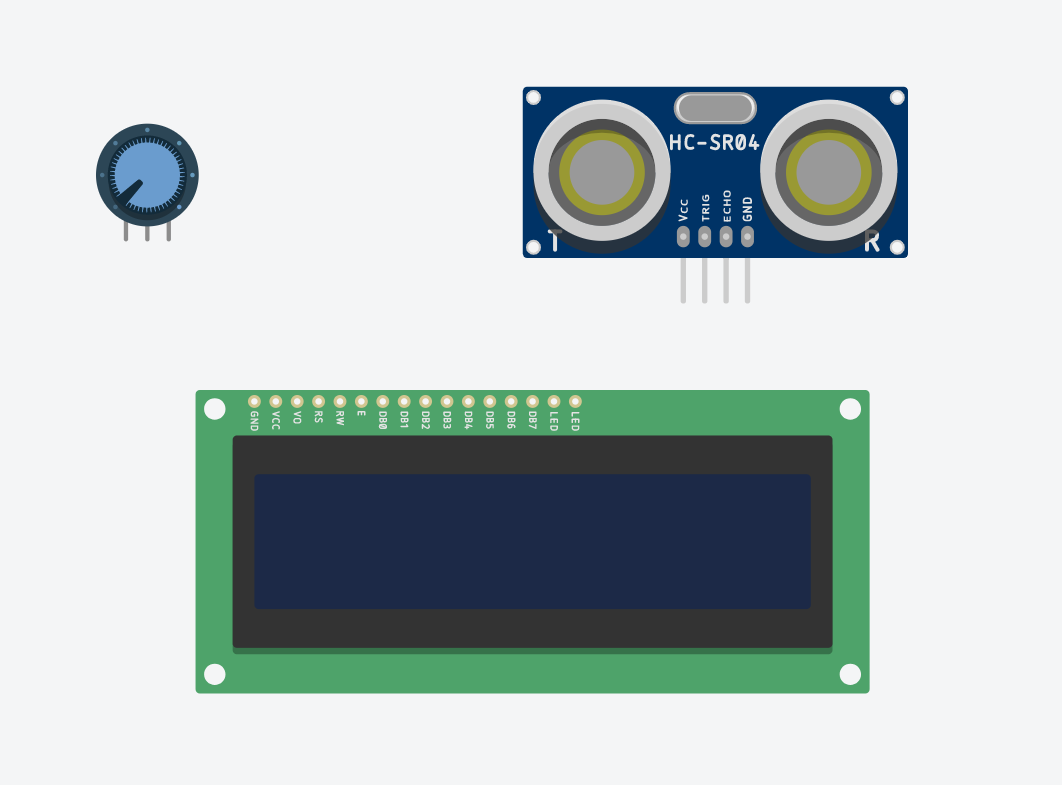
These LCDs are available in many different sizes (16×2 1602, 20×4 2004, 16×1 etc.), but they all use the same HD44780 parallel interface LCD controller chip from Hitachi. This means you can easily swap them. The LCD has 16 connection pins, numbered 1-16 from left to right. If the display does not include a resistor, you will need to add one between 5 V and pin 15. It should be safe to use a 220Ω resistor, but this value might make your display a bit dim. In this experiment, we used a potentiometer to get the best brightness. The maximum current rating of the backlight can be checked from the and used this to select an appropriate resistor value.[3]



**FIGURE 3:** LIQUID CRYSTAL DISPLAY

1. Potentiometer:

The potentiometer, commonly referred to as a “pot”, is a three-terminal mechanically operated rotary analogue device which can be found and used in a large variety of electrical and electronic circuits. They are passive devices, meaning they do not require a power supply or additional circuitry in order to perform their basic linear or rotary position function.[4] In this experiment, we will use the potentiometer to get the best brightness of the LCD screen available.



**FIGURE 4:** POTENTIOMETER

1. Piezo Buzzer:



FIGURE 5: PIEZO BUZZER

Piezo buzzers are simple devices that can generate basic beeps and tones. They work by using a piezo crystal, a special material that changes shape when voltage is applied to it. Simple change the frequency of the voltage sent to the piezo and it will start generating sounds by changing shape very quickly.[5] In this experiment, the Piezo buzzer will be used as a substitute for the fan due to simulation limitations in TinkerCAD.

* + 1. **Code Components**

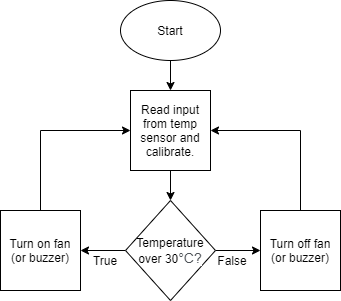
1. const: A variable qualifier that modifies the behavior of the variable, making a variable "read-only" and constant.
2. void setup: The function is called whenever the program starts. It is used to initialize variables, pin modes, etc. It will only run once after each powerup or reset of the Arduino board.
3. pinMode(): A function used to configure a specific pin to behave either as an input or an output, usually in the void setup.
4. void loop: The function that holds the code inside and runs over and over as long as the board is turned on.
5. LiquidCrystal lcd: Provides a set of endpoints to manage Arduino IoT Cloud Devices, Things, Properties and Timeseries. This API can be called just with any HTTP Client, or using one of these clients
6. lcd.setCursor(): Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
7. lcd.print(): Displays what is in the argument on the LCD Screen
8. tone(): Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.
   1. **Methods**
      1. **Code Description**

Once the Arduino Uno turns on, it sets up the pins for the LCD display (in our case pins 2, 3, 4, 5, 6, and 7 as the RS, enable, D4, D5, D6, and D7 respectively) to be able to print the output on it. And sets up the pin 8 for the buzzer that’s acting as the fan in this experiment, and the analog pin A0 for the TMP36 sensor as an analog input to be able to read the different voltages and get their corresponding temperatures.

In the void loop, the Arduino reads and stores the analog input from the temperature sensor at pin A0. The value stored is in millivolts, but since the sensor is linearly proportional to the temperature in degree Celsius, a few mathematical operations would calculate the current temperature to a certain accuracy by using equation 1, where the voltage is the reading from the sensor.

The voltage is divided by 1024 since the analog input gives us a value between 0 and 1023, 0 being no voltage and 1023 being 5V, so dividing by 1024 gives the percentage in this range. So, to find the right voltage for the current temperature, we multiply it by 5V since the sensor is powered on by the 5V pin. Since we are not measuring 0° – 175° we will need to shift the output value so that the minimum reading of -50° equals a voltage reading of 0. We do this by subtracting 0.5 from the output voltage. Our new value now ranges from -0.5 to 1.25. And finally, to convert from millivolts to degrees Celsius, we will multiply it by 100.

For the fan (or buzzer), once the final temperature calculated in degrees Celsius reaches or exceeds the threshold of 30°C, the fan would turn on, otherwise, would remain or turn back off as shown in the flowchart in figure 6.

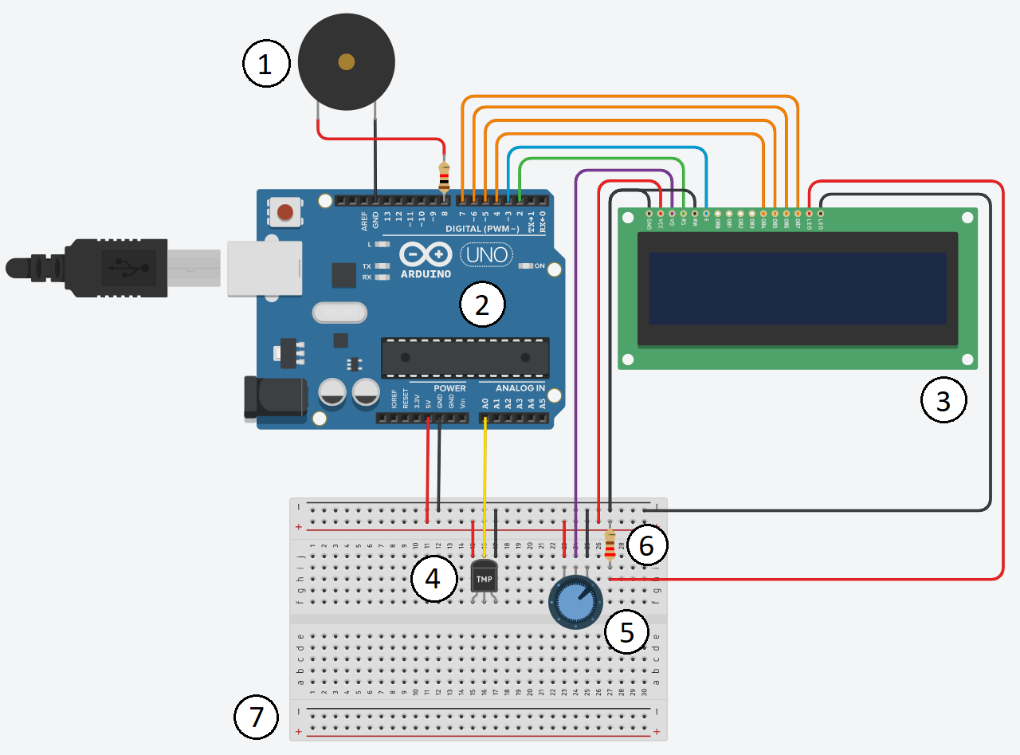


**FIGURE 6:** FLOWCHART

* + 1. **Connections on Tinker CAD**

1. First, connect the 5V and ground pins of the Arduino Uno to the breadboard power and ground (red and black wires).
2. For the LCD, connect the data pins D4, D5, D6, and D7 to the Arduino at pins 4, 5, 6, and 7 respectively to be able to print on the display (orange wires). The ground, R/W, and cathode pins are connected to the ground pin of the breadboard, and both the VCC and anode of the LCD are connected to the 5V of the breadboard, however, the anode is first connected to a 220Ω resistor to avoid damaging the screen.
3. The RS pin of the LCD is connected to pin 2 (green wire), and the Enable to pin 3 (blue wire), and finally V0 is connected to a potentiometer (purple wire) which is connected to the 5V and ground of the breadboard.
4. For the buzzer, the positive side is connected to a 500Ω resistor which is then connected to pin 7 of the Arduino (red wire), and the negative side is connected to the ground of the Arduino through the breadboard (black wires).
5. Finally, for the TMP36 temperature sensor, connect the power and ground pins to the 5V and ground of the Arduino (red and black wires). To receive the different temperature readings, connect the VOUT pin to the analog pin A0 of the Arduino (yellow wire).
6. **RESULTS AND DISCUSSION**
   1. **Simulation**

We implemented the circuit on Tinker CAD, adding the code to the Arduino Simulation, obtaining the circuit shown in Figure 7.



**FIGURE 7:** CIRCUIT SCHEMATIC ON TINKER CAD

|  |  |
| --- | --- |
| NUMBER | COMPONENT |
| 1 | Piezo buzzer (or fan) |
| 2 | Arduino Uno Board |
| 3 | 16x2 Liquid Crystal Display |
| 4 | TMP36 Temperature Sensor |
| 5 | Potentiometer |
| 6 | 220Ω Resistor |
| 7 | Breadboard |

**TABLE 1:** LEGEND

* 1. **Analysis**

After running the simulation, the LCD screen immediately printed the current temperature in Celsius based on the voltage from the TMP36 sensor and performing some mathematical operations. As the temperature measured by the sensor increased, the voltage increased, and the temperature printed on the LCD increased as well. As long as the temperature was lower than 30°C, the fan (or buzzer) remained off, once the temperature reached or exceeded this threshold, the buzzer turned on. The sensor had a range of −40°C to +125°C with very minimal errors. A video of the simulation can be shown through [this link](https://drive.google.com/file/d/1h1rbmdfLH8-QRf-wxiLeEzXKEzdf7_3g/view?usp=sharing).

1. **CONCLUSION**

The objective of this experiment is to measure the temperature using a TMP36 sensor and convert the analog signal to a digital output and display the correct temperature on an LCD screen in degrees Celsius. A fan (or buzzer) turns on once the calculated temperature exceeded the threshold of 30°C, and remains off otherwise. The experiment was successful, and the simulation behaved exactly like we programmed it to.

**REFERENCES**

[1] Pololu Robotics & Electronics, *Arduino Uno*, 2020

https://www.pololu.com/product/2191#:~:text=Overview,header%2C%20and%20a%20reset%20button.

[2] Analog Devices, *TMP36*

https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35\_36\_37.pdf

[3] Maker Guides, *How to use a 16×2 character LCD with Arduino*, 2020

https://www.makerguides.com/character-lcd-arduino-tutorial/

[4] Electronics Tutorials, *Potentiometers*, 2020

https://www.electronics-tutorials.ws/resistor/potentiometer.html

[5] Adafruit, *Using Piezo Buzzers with CircuitPython & Arduino*, 2018

https://learn.adafruit.com/using-piezo-buzzers-with-circuitpython-arduino

[6] Autodesk Tinkercad

https://www.tinkercad.com/

**Appendix**

* Arduino Code:

#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

const int buzzer = 8;

float temperature;

void setup() {

lcd.begin(16, 2);

pinMode(buzzer, OUTPUT);

}

void loop() {

temperature = analogRead(A0); //read from the analog sensor and store it

temperature = temperature / 1024; //find percentage of input reading

temperature = temperature \* 5; //multiply by 5V to get voltage

temperature = temperature - 0.5; //subtract the offset

temperature = temperature \* 100; //convert to degrees Celsius

lcd.setCursor(0, 0);

lcd.print("Temp: ");

lcd.print(temperature);

lcd.setCursor(15,0);

lcd.print("C");

if(temperature >= 30)

tone(buzzer,600);

else

noTone(buzzer);

}