**Experiment IOT MID Term Lab**

# **Aim:** **Automatic Room Temperature Controller**

**Apparatus**:LCD 16\*2, Arduino Uno .Connecting Wire, LED ,General Purpose Transistor NPN,temperature sensor,1N4007 – High Voltage, High Current Rated Diode.

**Theory:-**Weather changes become hard to adapt. That is, during Winter we face difficulties tolerating the freezing cold, and that is why people often prefer wearing coats during the season. On the other hand, the weather becomes too warm in summer. Thus, having understood the switching operation of transistors, unidirectional current flow in diodes, the principle of operation of motors, the resistance from resistors combined with the transformation capability of the transducer, the temperature sensor in this case, I would like to summarize the operation of my project as follows.

1. If the temperature exceeds the maximum of the aforementioned "desired" range, then the LCD displays that the temperature is higher and informs the FAN to turn on. Then the FAN starts its rotation/vibration, and after a while the temperature gets lowered falling in the range, then the LCD commands the FAN to turn off.

2. Whenever the sensor's temperature reading goes down below the possible minimum temperature in the range, the LCD notifies that the temperature is LOWER and tells the heater to be turned on, and after the temperature is in the range, it displays that temperature is OK and orders the heater to be switched off.

3. The last condition is whenever the temperature is within the desired range, the LCD tells that the temperature is normal. Thus, it asks to turn off everything. This condition can, for instance, be observed if you run the code and the slider's position remains unchanged. That tells one that the default slider position is within the desired room temperature range (i.e. the default is 24.78 degree C).

**1N4007 – High Voltage, High Current Rated Diode**

**-1N4007** is a rectifier **diode**, designed specifically for circuits that need to convert alternating **current** to direct **current**. It can pass currents of up to 1 A, and have peak inverse **voltage** (PIV) **rating** of 1,000 V

**Temperature sensor-**

A **temperature sensor** is a device, usually an RTD (resistance **temperature detector**) or a thermocouple, that collects the data about **temperature** from a particular source and converts the data into understandable form for a device or an observer.

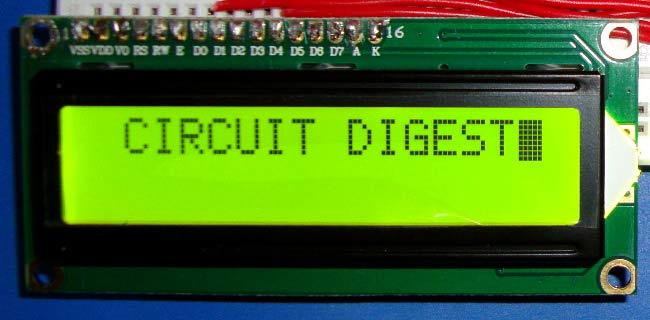
fig-temperature sensor

**[NPN](https://en.wikipedia.org/wiki/NPN_transistor" \o "NPN transistor) [bipolar junction transistor](https://en.wikipedia.org/wiki/Bipolar_junction_transistor" \o "Bipolar junction transistor)-**

The **2N2222** is a common [NPN](https://en.wikipedia.org/wiki/NPN_transistor" \o "NPN transistor) [bipolar junction transistor](https://en.wikipedia.org/wiki/Bipolar_junction_transistor" \o "Bipolar junction transistor) (BJT) used for general purpose low-power [amplifying](https://en.wikipedia.org/wiki/Amplifier" \o "Amplifier) or switching applications. It is designed for low to medium [current](https://en.wikipedia.org/wiki/Electric_current" \o "Electric current), low [power](https://en.wikipedia.org/wiki/Electric_power" \o "Electric power), medium [voltage](https://en.wikipedia.org/wiki/Voltage" \o "Voltage), and can operate at moderately high speeds. It was originally made in the [TO-18](https://en.wikipedia.org/wiki/TO-18" \o "TO-18) metal can as shown in the picture.

The 2N2222 is considered a very common transistor,and is used as an exemplar of an NPN transistor. It is frequently used as a small-signal transistor,and it remains a small general purpose transistor of enduring popularity.

# [16\*2 LCD Display Module](https://circuitdigest.com/article/16x2-lcd-display-module-pinout-datasheet)-

16x2 LCD Display Module with HD44780 Controller

****16×2 LCD**** is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. But the most used one is the 16\*2 LCD, hence we are using it here.

All the above mentioned LCD display will have 16 Pins and the programming approach is also the same and hence the choice is left to you. Below is the ****Pinout and Pin Description of 16x2 LCD Module****:

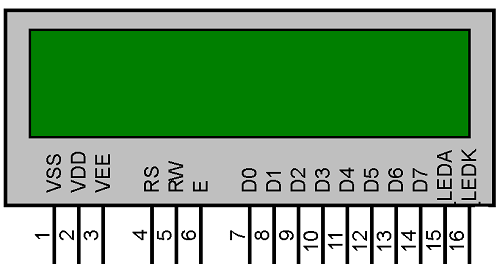


Fig1.LED 16\*2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ****Sr. No**** | ****Pin No.**** | ****Pin Name**** | ****Pin Type**** | ****Pin Description**** | ****Pin Connection**** |
| 1 | Pin 1 | Ground | Source Pin | This is a ground pin of LCD | Connected to the ground of the MCU/ Power source |
| 2 | Pin 2 | VCC | Source Pin | This is the supply voltage pin of LCD | Connected to the supply pin of Power source |
| 3 | Pin 3 | V0/VEE | Control Pin | Adjusts the contrast of the LCD. | Connected to a variable POT that can source 0-5V |
| 4 | Pin 4 | Register Select | Control Pin | Toggles between Command/Data Register | Connected to a MCU pin and gets either 0 or 1. 0 -> Command Mode 1-> Data Mode |
| 5 | Pin 5 | Read/Write | Control Pin | Toggles the LCD between Read/Write Operation | Connected to a MCU pin and gets either 0 or 1. 0 -> Write Operation 1-> Read Operation |
| 6 | Pin 6 | Enable | Control Pin | Must be held high to perform Read/Write Operation | Connected to MCU and always held high. |
| 7 | Pin 7-14 | Data Bits (0-7) | Data/Command Pin | Pins used to send Command or data to the LCD. | In 4-Wire Mode Only 4 pins (0-3) is connected to MCU In 8-Wire Mode All 8 pins(0-7) are connected to MCU |
| 8 | Pin 15 | LED Positive | LED Pin | Normal LED like operation to illuminate the LCD | Connected to +5V |
| 9 | Pin 16 | LED Negative | LED Pin | Normal LED like operation to illuminate the LCD connected with GND. | Connected to ground |

These black circles consist of an interface IC and its associated components to help us use this LCD with the MCU. Because our LCD is a 16\*2 Dot matrix LCD and so it will have (16\*2=32) 32 characters in total and each character will be made of 5\*8 Pixel Dots.  A Single character with all its Pixels enabled is shown in the below picture.

So Now, we know that each character has (5\*8=40) 40 Pixels and for 32 Characters we will have (32\*40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels.

It will be a hectic task to handle everything with the help of MCU, hence an ****Interface IC like HD44780**** is used, which is mounted on LCD Module itself. The function of this IC is to get the ****Commands and Data**** from the MCU and process them to display meaningful information onto our LCD Screen.

Let’s discuss the different type of mode and options available in our LCD that has to be controlled by our Control Pins.

**4-bit and 8-bit Mode of LCD:**

The LCD can work in two different modes, namely the 4-bit mode and the 8-bit mode. In ****4 bit mode**** we send the data nibble by nibble, first upper nibble and then lower nibble. For those of you who don’t know what a nibble is: a nibble is a group of four bits, so the lower four bits (D0-D3) of a byte form the lower nibble while the upper four bits (D4-D7) of a byte form the higher nibble. This enables us to send 8 bit data.

Whereas ****in 8 bit mode**** we can send the 8-bit data directly in one stroke since we use all the 8 data lines.

Now you must have guessed it, Yes 8-bit mode is faster and flawless than 4-bit mode. But the major drawback is that it needs 8 data lines connected to the microcontroller. This will make us run out of I/O pins on our MCU, so 4-bit mode is widely used. No control pins are used to set these modes. It's just the way of programming that change.

**Read and Write Mode of LCD:**

As said, the LCD itself consists of an Interface IC. The MCU can either read or write to this interface IC. Most of the times we will be just writing to the IC, since reading will make it more complex and such scenarios are very rare. Information like position of cursor, status completion interrupts etc. can be read if required, but it is out of the scope of this tutorial.

The Interface IC present in most of the LCD is ****HD44780U,****in order to program our LCD we should learn the complete datasheet of the IC. The datasheet is given here.

**LCD Commands:**

There are some preset commands instructions in LCD, which we need to send to LCD through some microcontroller. Some important command instructions are given below:

|  |  |
| --- | --- |
| ****Hex Code**** | ****Command to LCD Instruction Register**** |
| 0F | LCD ON, cursor ON |
| 01 | Clear display screen |
| 02 | Return home |
| 04 | Decrement cursor (shift cursor to left) |
| 06 | Increment cursor (shift cursor to right) |
| 05 | Shift display right |
| 07 | Shift display left |
| 0E | Display ON, cursor blinking |
| 80 | Force cursor to beginning of first line |
| C0 | Force cursor to beginning of second line |
| 38 | 2 lines and 5×7 matrix |
| 83 | Cursor line 1 position 3 |
| 3C | Activate second line |
| 08 | Display OFF, cursor OFF |
| C1 | Jump to second line, position 1 |
| OC | Display ON, cursor OFF |
| C1 | Jump to second line, position 1 |
| C2 | Jump to second line, position 2 |

LED-A **light-emitting diode** (**LED**) is a [semiconductor](https://en.wikipedia.org/wiki/Semiconductor" \o "Semiconductor) [light source](https://en.wikipedia.org/wiki/Light_source" \o "Light source) that emits light when [current](https://en.wikipedia.org/wiki/Electric_current" \o "Electric current) flows through it. [Electrons](https://en.wikipedia.org/wiki/Electron" \o "Electron) in the semiconductor recombine with [electron holes](https://en.wikipedia.org/wiki/Electron_hole" \o "Electron hole), releasing energy in the form of [photons](https://en.wikipedia.org/wiki/Photon" \o "Photon). The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the [band gap](https://en.wikipedia.org/wiki/Band_gap" \o "Band gap) of the semiconductor.[[5]](https://en.wikipedia.org/wiki/Light-emitting_diode" \l "cite_note-5) White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.[[6]](https://en.wikipedia.org/wiki/Light-emitting_diode" \l "cite_note-6)



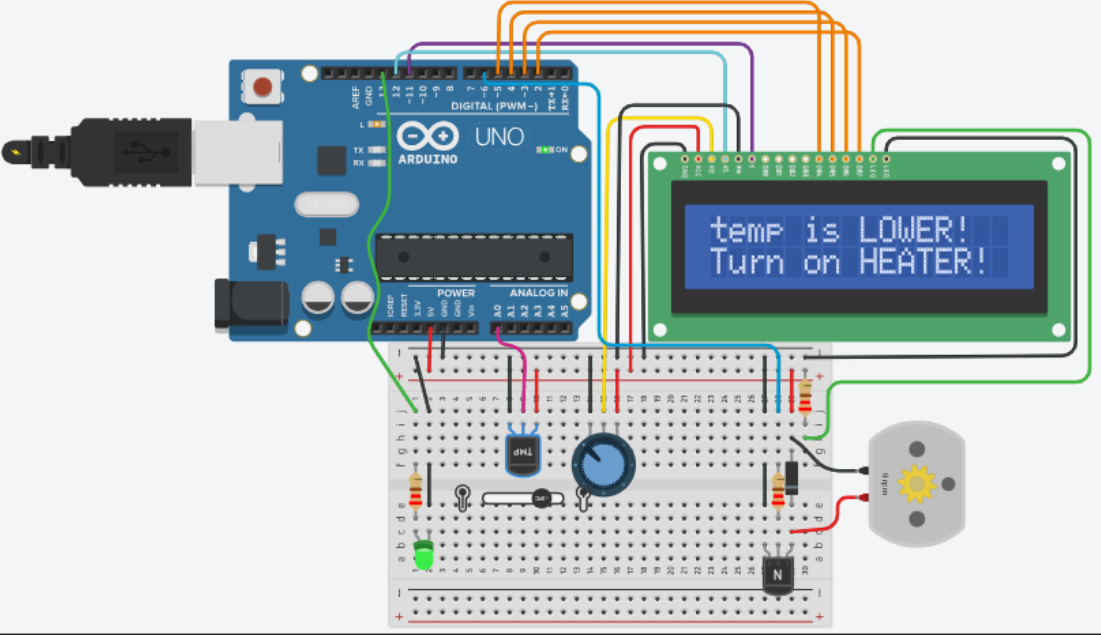
Fig 2: LED

Arduino UNO-The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which six can be used as PWM outputs), six analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an 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.

The Arduino Uno differs from all preceding boards because it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega8U2 programmed as a USB-to-serial converter. Revision 2 of the Arduino Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

|  |  |
| --- | --- |
| ****Features**** |  |
| * 14 digital I/O pins (six of which provide PWM output) * 3.3 V supply generated by an on-board regulator * Six analog input pins * 32 KB of flash memory | * Can supply 40 mA of DC current per pin * 16 MHz clock speed * Code example from Arduino website to help get started |

**Circuit Diagram:**



**CODE-**

const int temp\_trans\_pin = A0, Heater\_pin = 13, FAN\_pin = 6;

float MinTemp = 20, MaxTemp = 25;

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

void setup() {

LCD.begin(16, 2);

pinMode(Heater\_pin, OUTPUT);//LED in our case

pinMode(FAN\_pin, OUTPUT);

LCD.print("Room temp(C):");

LCD.setCursor(2,1);

LCD.print(MinTemp); LCD.print("-");LCD.print(MaxTemp);

delay(2000);

}

void loop() {

float Eqv\_volt, SensorTemp;

Eqv\_volt = analogRead(temp\_trans\_pin) \* 5.0 / 1023;

SensorTemp = 100.0 \* Eqv\_volt-50.0;

LCD.clear();

LCD.print("Sensor reading:");

LCD.setCursor(2,1);

LCD.print(SensorTemp); LCD.print(" C");

delay(2000);

if(SensorTemp > MaxTemp){

LCD.clear();

LCD.print("temp is HIGHER!");//higher than the max

/\*Turn on FAN (dc motor)! to regulate the temp.

Increase FAN speed at a slow rate\*/

LCD.setCursor(0, 1);LCD.print("Turn on FAN!");

for( int i = 0; i <= 255; i++ ) {

analogWrite(FAN\_pin, i);

}

delay(2000);

LCD.clear();

LCD.print("Now temp is OK!");

LCD.setCursor(0, 1);

LCD.print("Turn off FAN!");

for( int i = 255; i >= 0; i-- ) {

analogWrite(FAN\_pin, i);

}

delay(2000);

}

else if(SensorTemp < MinTemp){

LCD.clear();

LCD.print("temp is LOWER!");//Less than the mini

LCD.setCursor(0, 1);

LCD.print("Turn on HEATER!");

//Turn the heater ON, LED in our case

digitalWrite(Heater\_pin, HIGH);

delay(3000);

LCD.clear();

LCD.print("Now temp is OK!");

LCD.setCursor(0, 1);

LCD.print("Turn off HEATER!");

delay(1000);

digitalWrite(Heater\_pin, LOW);

LCD.clear();

}

else if(SensorTemp > MinTemp && SensorTemp < MaxTemp)

LCD.clear();

LCD.print("Temp is NORMAL!");LCD.setCursor(2,1);

LCD.print("Turn off all!");

delay(1000);

LCD.clear();

}

else {

LCD.clear();

LCD.print("Something went");

LCD.setCursor(2,1); LCD.print("WRONG in the ckt");

delay(1000);

LCD.clear();

}

delay(1000);

}

Working :-Arduino Programming and Hardware Fundamentals . As it can be seen from the design, it consists of the LCD, the heater (LED), FAN (DC motor) as its main output blocks, and the potentiometer and temperature sensor (TMP36) as the input control blocks. In addition, it contains the signal conditioning and component protection elements such as a transistor, diode, and the resistors. Here the temperature of a room is needed to be in [20, 25] Celsius.  
Once the tinkerCAD circuitry is accessed one can execute the code by pressing on the button named "Start Simulation". Then while it's running, the person needs to move the slider, which appears whenever a mouse click is made on the temperature sensor (TMP36) so as to simulate the temperature variations. Then either the heater (LED) or the FAN (DC motor) responds to the changes brought by the slider, and the LCD displays the changes and the sensor readings step by step. Thus, a brief summary of the controller is as follows:  
1. If the temperature exceeds the maximum of the aforementioned "desired" range, then the LCD displays that the temperature is higher and tells to turn on the FAN. Then the FAN starts its rotation, and after the temperature is in the range the LCD order the FAN to turn off.  
2. Whenever the sensor's temperature reading goes down below the minimum temperature in the range, the LCD notifies that the temperature is LOWER and tells the heater to be turned on, and after the temperature is in the range it displays that it is ok and orders the heater to be switched off.  
3. The last condition is whenever the temperature is within the desired range, the LCD tells that the temperature is normal. Thus, it asks to turn off everything. This condition can, for instance, be observed if you run the code and wait without affecting the slider's position. That tells one that the default slider position is within the desired room temperature range (i.e. the default is 24.78 dg C).  
System iterates forever unless it gets terminated by the user due to the behavior of the loop ( ) function.