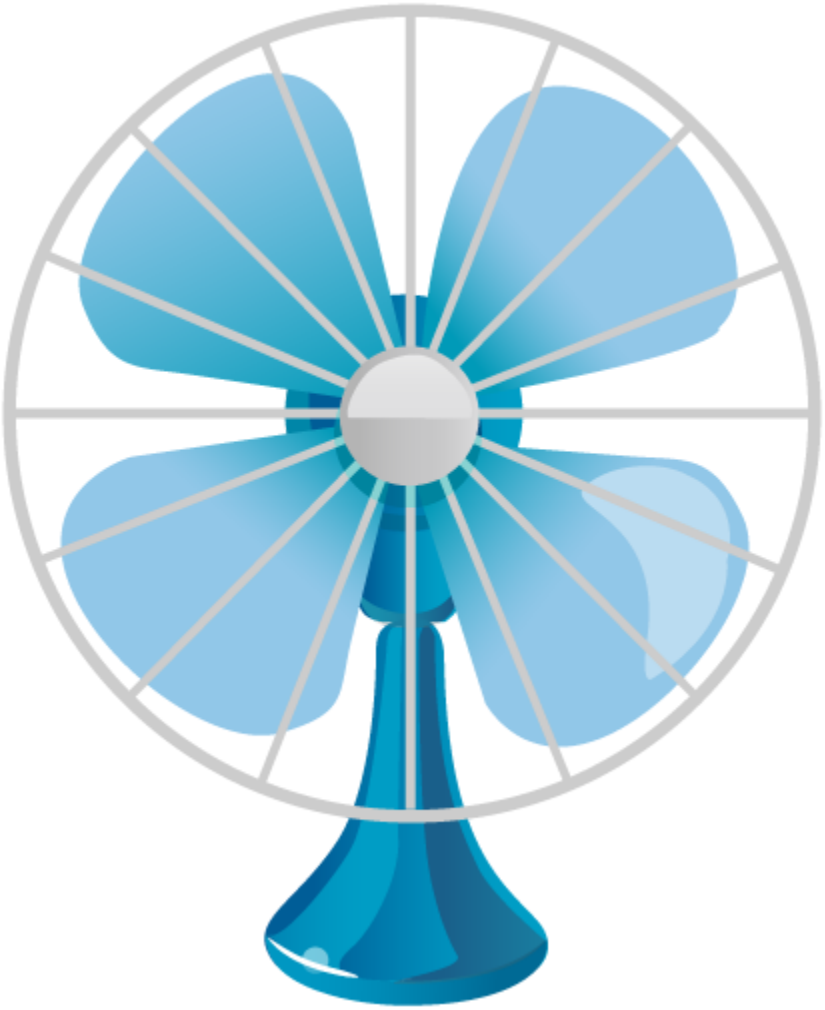
Self-Cooling Temperature Regulation System



By: Jake Bastin

Table of Contents

Overview of Device

My Arduino Device is a self-cooling temperature regulation system that uses a fan to cool down an area and warn users if the temperature becomes “extreme”. I came up with this device by thinking about how nice it would be to have a thermostat/cooling system that alerts me when the temperature is too hot (extreme temperature regulation) in order for me to take immediate action. The idea of a self-reliant system that automatically cools down the area without any manual activation is also a bonus for those who are unable to do anything. An LCD screen that displays both the temperature in Celsius and Fahrenheit make it possible to be used in nations that use the imperial, as well as metric system. As global warming is impacting the world, temperatures are beginning to rise, and not in small amounts. As such, having such a system will be crucial in areas that are not prone to heat waves and humid weather.

The prototype design for this device was created using the Arduino microcontroller as well as a small D.C. motor/fan combo and a thermistor sensor. It is very simple to set up and easy to use. Wiring connects the Arduino board to the LCD display and is also connected to the thermistor. The D.C. motor with a fan is also attached and is powered in series to a buzzer. This way, the buzzer activates at the same time as the motor.

The thermistor is constantly reading the ambient temperature of its surroundings. If the temperature exceeds 30 °C, the thermistor will relay this information to the LCD and the D.C. motor. The computer program written will then send out a pin voltage output from the Arduino to the D.C. motor as well as the buzzer. At this point, the buzzer will start to beep and the D.C. motor will spin an attached fan. The wind current produced by the fan will cool down the surrounding environment and stabilize the temperature. Once the temperature has reached an optimal state, the fan and buzzer will deactivate to indicate that no action should be taken at the moment.

Safety

1.) Do not tamper with the wiring of the circuit. Tampering with the wiring may also damage the circuit and cause the system to malfunction (there goes your hard-earned cash).

2.) Although the circuit does not contain lethal amounts of current, there is still a possibility of small injury from electric shock if you mess with the device too much.

3.) Keep away from children! There are small parts that pose as a choking hazard to young kids. Some exposed wires that may be present are dangerous to young kids since they tend to chew on things. Don't be a Yahoo! Keep the device secure at all times.

4.) During installation, keep liquids and conductive substances AWAY from the device as it is an electrical hazard! Keep the fan unobstructed as the device may overheat otherwise.

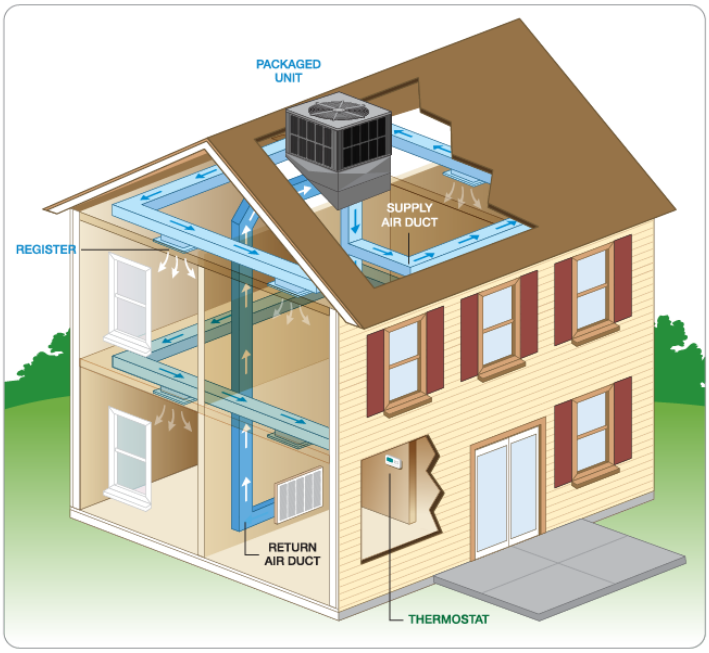
5.) Once the device has been installed, there is no need to tamper with the device or take it down. It is advisable to install it in a desirable location up installation in order to minimized contact with the device. This is to minimize the damage/impact to the minuscule components present in the circuitry.

Instructions

To build this device, you need 1 Arduino board and the parts listed on pages 7 - 8. The base of this device will be the breadboard. The breadboard is where all of the circuitry will go. The wiring of the circuitry is displayed on page 9. The LCD1602 display will be inserted into the breadboard and attached to the Arduino through the wiring shown on page 9. Insert a potentiometer to contain the current flow to the LCD and calibrate it until the display's text is visible. Insert the buzzer and thermistor components and wire them as shown in the diagram. Finally, attach the small fan to the top of the D.C. motor and wire the motor as shown. Once the wiring and calibration are complete, copy and paste the code from pages 10 - 11 into the Arduino IDE on your computer. Plug the Arduino board into your computer via the USB cable and upload the program to your board.

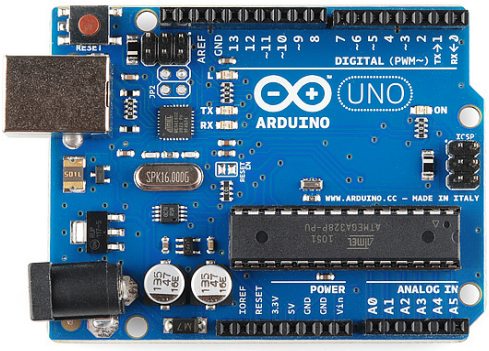
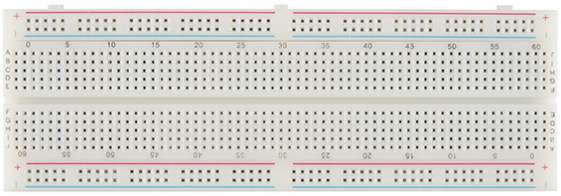
In order to verify that the device is functioning properly, first check if the LCD display is showing the temperature of the room in Celsius as well as Fahrenheit. If so, the LCD display is functioning properly. Secondly, place your finger on the thermistor to increase the heat transferred, if the LCD screen displays an increase in temperature, then the thermistor is functioning as it should. At this point, observe if the fan begins to spin and the buzzer starts to make noise (if the temperature is greater than 30°C). If so, the fan and buzzer are functioning correctly. Finally, wait for the temperature to drop below 30 °C and return to room temperature. If the buzzer and fan deactivate, this ensures that the code running the device is correct. Place the device in an area (preferably high up with no obstruction) where you would like to monitor the temperature. Note: The temperature at which the device activates can be adjusted by the user by editing the source code! If all of these steps are done correctly, the device is working properly and you can now enjoy your self-cooling temperature system!

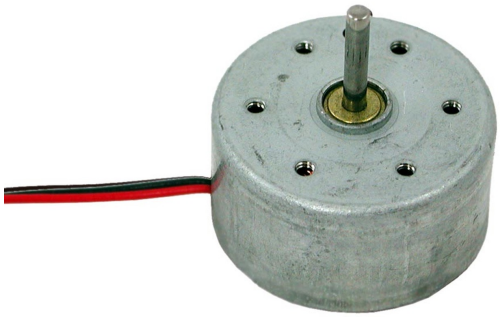
Research



I initially got my inspiration from the design of the system shown in the above image. This is a home cooling system that must be manually configured to stabilize the temperature in a house. My invention improves upon this design by adding in an LCD display that is constantly updating the ambient temperature every 100 milliseconds which is much faster than a traditional thermostat. The added bonus of a self-activating fan makes it convenient and the buzzer is important to alert users to any dangers. I decided to combine the buzzer, LCD and D.C Motor/Fan as I was completing my Arduino Labs where these components were used independently. By combining them, I was able to create something that is more or less revolutionary for the future of climate change and house appliances. Although the device may seem simple, this is exactly one of its benefits. By using these sensors and my already working knowledge of them, I was able to create something that is robust and not complex making it great for user comprehension. I used these 4 components because they seemed the most interesting to me and were highly vital in designing a convenient device.

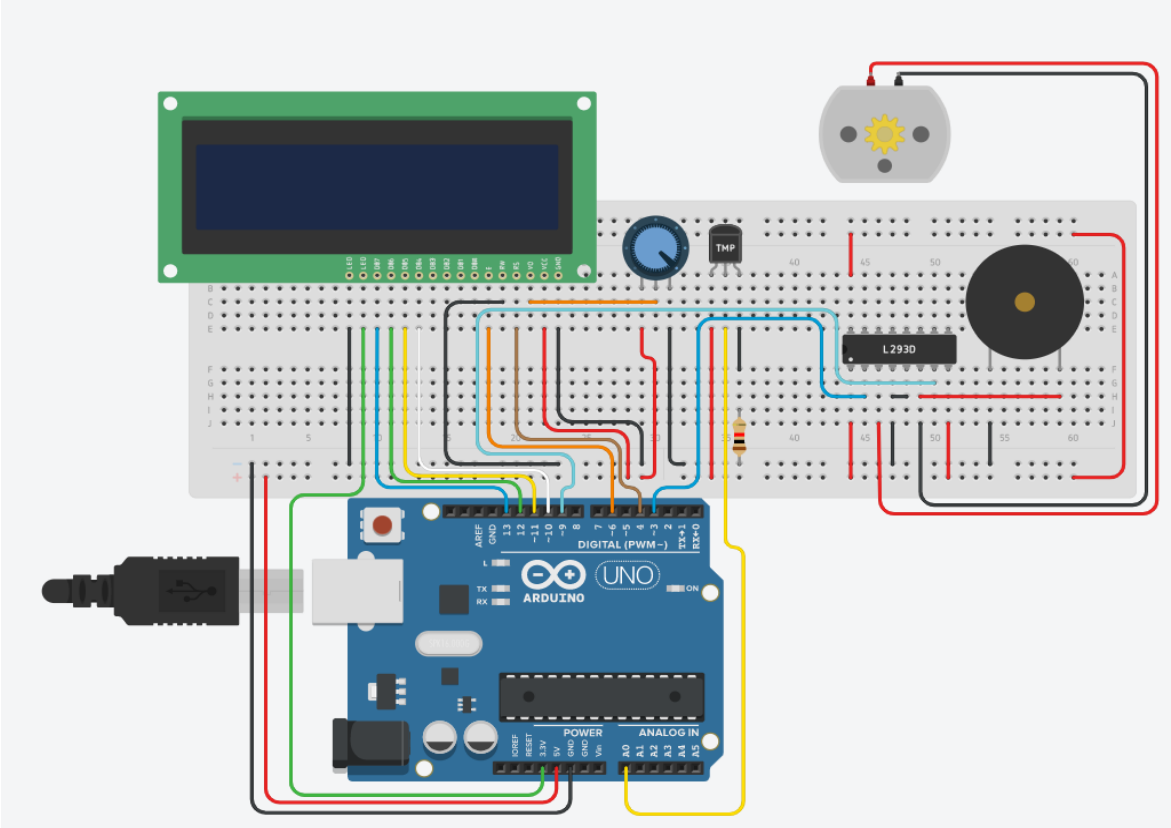
Parts Required

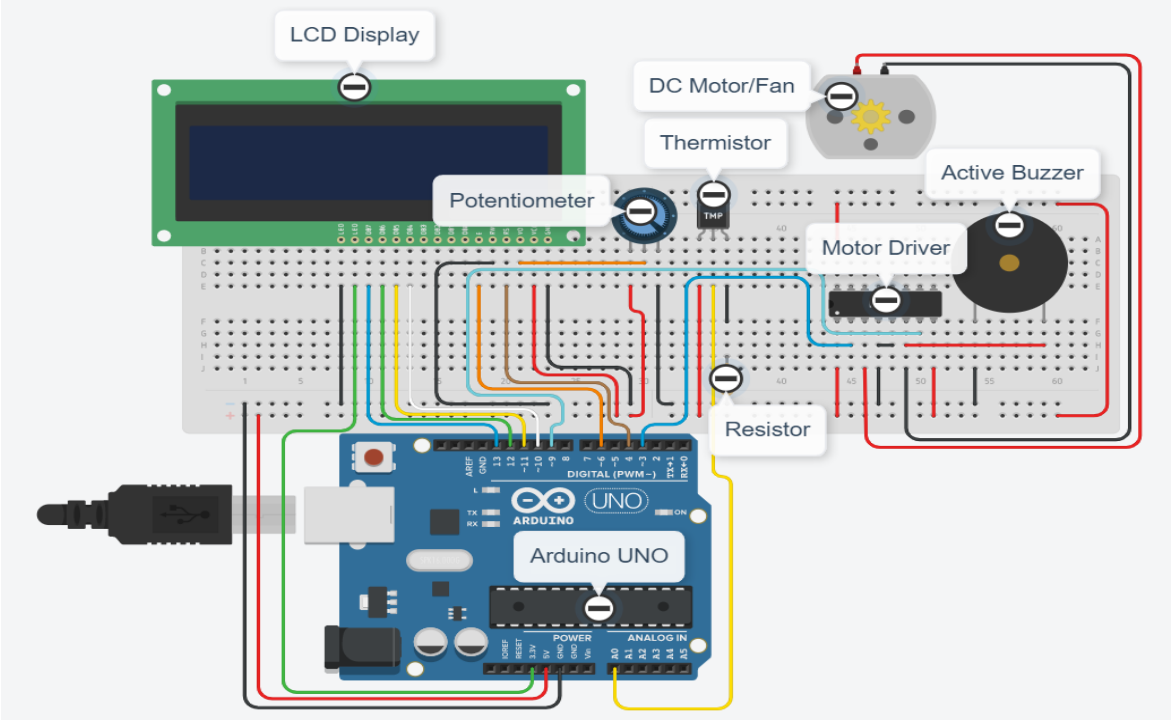
Component	Appearance
Arduino Uno (x1)	
USB Cable 2.0 (x1)	
Standard Breadboard (x1)	

Assorted Jumper Wires (x32)	
LCD1602 (x1)	
D.C. Motor (x1)	
Plastic Fan Attachment (x1)	

Thermistor (x1)	
Active Buzzer (x1)	
L293D Motor Driver IC (x1)	
Potentiometer (x1)	
220 Ω Resistor (x1)	

Circuit Diagrams





Code Breakdown

Code	*Function*

`#include <LiquidCrystal.h>`
 	
`LiquidCrystal lcd(4, 6, 10, 11, 12, 13);`
 	
`#define analogPin A0`
`#define beta 3950`
`#define resistance 10`
`#define rotation 250`
	This code imports the LCD library necessary for the program to interact with the LCD1602 component and initializes it.
	
	The following "define" code assigns each of the variables to various things such as the analog pin and rotation speed value for the D.C. motor.

`const int motor1 = 9;` `const int motor2 = 3;`	This code assigns the inputs of the D.C. motor to pins 9 and 3 of the Arduino board.

`void setup()`
`{`
 	
`lcd.begin(16, 2);`
`lcd.clear();`
 	
`pinMode(motor1,OUTPUT);`
`pinMode(motor2,OUTPUT);`
`Serial.begin(9600);`
 	
`}`
	This is the setup method of an Arduino sketch. This is where code that runs on startup is placed.
	
	In this method, the LCD is completely cleared.
	
	This code refers back to the variables where pins 9 and 3 were stored and sets both of those pins to outputs. This allows the Arduino to output signals from these pins.

`void loop()`
`{`
 	
`long thermistorValue = analogRead(analogPin);`
 	
`float celsius = beta / (log((1025.0 * 10 / thermistorValue - 10) / 10) + beta / 298.0) - 273.0;`
`float fahrenheit = 1.8 * celsius + 32.0;`
	The loop method is the code that runs while the program is running and loops for as long as the Arduino is running.
	
	This code sets the variable of the thermistor value to the value that the thermistor reads which is sent to the analog pin (A0), which was initialized and declared earlier.
	
	This code declares and initializes variables for celsius and fahrenheit and then converts the thermistor value to temperatures in °C and °F using mathematical calculations.

lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(celsius);
lcd.print(char(223));
lcd.print("C");
	This code sets the LCD cursor to the far left of the screen and prints out the temp in "Temp: " format. The variable celsius refers to the float declared earlier which displays the temp in °C.

lcd.setCursor(0, 1);
lcd.print("Fahr: ");
lcd.print(fahrenheit);
lcd.print(" F");
delay(100);
	This code sets the LCD cursor to the bottom of the celsius reading on the screen and then prints out the temp in "Fahr: " format. The variable fahrenheit refers to the float declared earlier which displays the temp in °F.
	
	The delay(100) delays the rate at which the readings change by 100 milliseconds.

if (celsius > 30.0)
{
fanSpin(rotation);
}
 	
else
{
fanSpin(0);
}
 	
}
	This if/else statement checks if the celsius variable that stores the thermistors reading is above 30°C. If it is, then the program calls to a method shown later which outputs a signal to the D.C. motor and spins the fan as well as powers the buzzer.
	
	If the temperature (celsius) is less than 30°C, the signal to the D.C. motor and buzzer is cut and both components are deactivated.

void fanSpin(int rotationSpeed)
{
 	
analogWrite(motor1,0);
analogWrite(motor2,rotationSpeed);
 	
}
	This is a method known as fanSpin that takes in the integer parameter of rotationSpeed which represents the speed at which the motor should rotate.
	
	When run, this method writes an analog signal to the inputs of the D.C. motor that spin the fan at the rotation speed declared earlier.

Full Code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(4, 6, 10, 11, 12, 13);

#define analogPin  A0
#define beta 3950
#define resistance 10
#define rotation 250

const int motor1 = 9;
const int motor2 = 3;

void setup()
{

  lcd.begin(16, 2);
  lcd.clear();

  pinMode(motor1,OUTPUT);
  pinMode(motor2,OUTPUT);
  Serial.begin(9600);

}

void loop()
{

  long thermistorValue = analogRead(analogPin);
  float celsius = beta / (log((1025.0 * 10 / thermistorValue - 10) / 10) + beta / 298.0) - 273.0;
  float fahrenheit = 1.8 * celsius + 32.0;

  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(celsius);
  lcd.print(char(223));
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("Fahr: ");
  lcd.print(fahrenheit);
  lcd.print(" F");
  delay(100);

  if (celsius > 30.0)
  {
    fanSpin(rotation);
  }

  else
  {
    fanSpin(0);
  }

}

void fanSpin(int rotationSpeed)
{

  analogWrite(motor1,0);
  analogWrite(motor2,rotationSpeed);

}
```

Reflection

In order to complete this project, I used the “Divide & Conquer” technique. Since I am using various components that each function differently, I had to individually consider how each component worked. I started off by wiring the LCD display first and adding the LCD configuration code. Then I added in the thermistor to output signals to the Arduino. Unfortunately, the pins for the Arduino were all filled up so to wire the D.C. motor, I had to configure a different pin for output and learn the coding for that. My original plan was to use the humiture sensor that displays the humidity factor and the temperature. Unfortunately for me, the sensor ended up malfunctioning and became faulty. Hence, I had to resort to using the thermistor sensor. The positive aspect of that was that the thermistor was really easy to wire and compact in my design. If I had to redo this assignment, I would definitely spend more time on it to make it more interesting as a project. I would try to add in a feature where the buzzer makes sounds at certain intervals to reduce annoying noises in a household. I would also add an LED light as a visual indicator (for impaired hearing). Additionally, I would also wish to make the LCD, display a message indicating that the temperature is too hot or some other type of customized message. Finally, I wish I was able to incorporate this system inside some sort of box or container in order to showcase the real-life capabilities of it instead of having just the raw circuit and wiring. On top of all this, having a user control the fan speed, activation temperature or manually activate the fan was also something I would hope to do during a second attempt. Overall I am fairly happy with my ability to combine the sensors and my practical understanding of them and develop a device (and its program) that may prove useful in the near future.

References

Lesson 12 LCD1602. sunfounder. (2020, October 30). https://learn.sunfounder.com/lesson-12-lcd1602/.

Lesson 22 Simple Creation - Small Fan. sunfounder. (2020, October 30). https://learn.sunfounder.com/lesson-22-simple-creation-small-fan/.

Simple Auto Fan. Arduino Project Hub. (n.d.). https://create.arduino.cc/projecthub/omer-beden/simple-auto-fan-4b2a50.