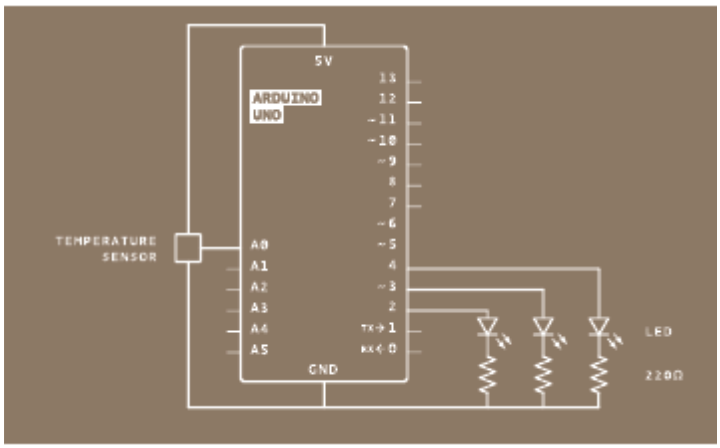
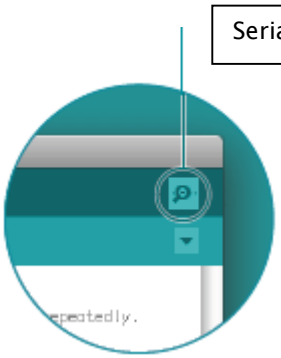
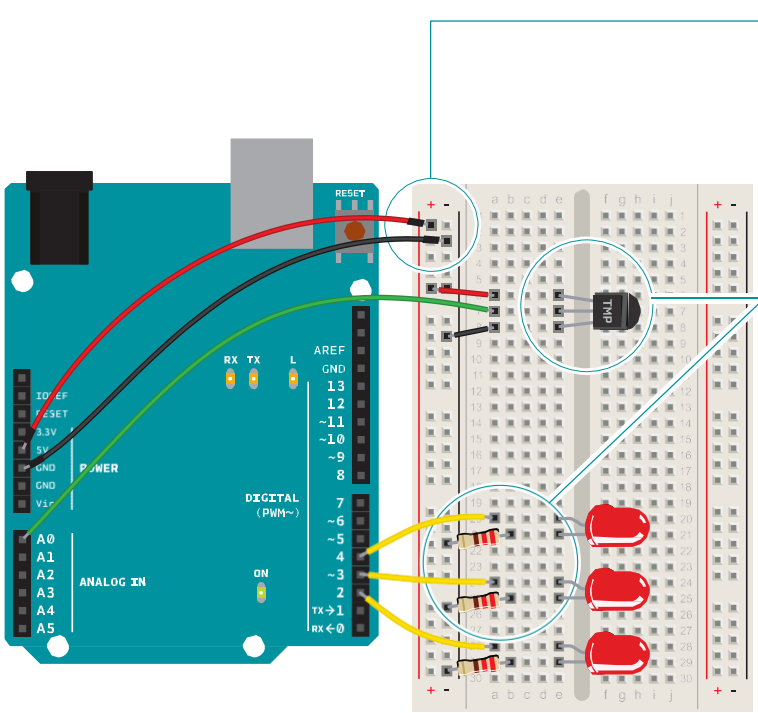
3. Analog sensors

You'll be using a temperature sensor to measure how warm your skin is. This component outputs a changing voltage depending on the temperature it senses. It has three pins: one that connects to ground, another that connects to power, and a third that outputs a variable voltage to your Arduino. In the sketch for this project, you'll read the sensor's output and use it to turn LEDs on and off, indicating how warm you are. The model TMP36 is convenient because it outputs a voltage that changes directly proportional to the temperature in degrees Celsius. The Arduino IDE comes with a tool called the *serial monitor* that enables you to report back results from the microcontroller. Using the serial monitor, you can get information about the status of sensors, and get an idea about what is happening in your circuit and code as it runs.







TMP 36:

Place it on the breadboard with the rounded part facing away from the Arduino (the order of the pins is important!) Connect the left pin of the flat facing side to power, and the right pin to ground. Connect the center pin to pin **A0** on your Arduino. This is analog input pin 0.

```
// named constant for the pin the sensor is connected to
const int sensorPin = A0;
// room temperature in Celsius
const float baselineTemp = 20.0;

void setup() {
  // open a serial connection to display values
  Serial.begin(9600);
  // set the LED pins as outputs
  // the for() loop saves some extra coding
  for (int pinNumber = 2; pinNumber < 5; pinNumber++) {
    pinMode(pinNumber, OUTPUT);
    digitalWrite(pinNumber, LOW);
  }
}

void loop() {
  // read the value on AnalogIn pin 0 and store it in a variable
  int sensorVal = analogRead(sensorPin);

  // send the 10-bit sensor value out the serial port
  Serial.print("sensor Value: ");
  Serial.print(sensorVal);

  // convert the ADC reading to voltage
  float voltage = (sensorVal / 1024.0) * 5.0;

  // Send the voltage level out the Serial port
  Serial.print(", Volts: ");
  Serial.print(voltage);

  // convert the voltage to temperature in degrees C
  // the sensor changes 10 mV per degree
  // the datasheet says there's a 500 mV offset
  // ((voltage - 500 mV) times 100)
  Serial.print(", degrees C: ");
  float temperature = (voltage - .5) * 100;
  Serial.println(temperature);

  // if the current temperature is lower than the baseline turn off all LEDs
  if (temperature < baselineTemp + 2) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  } // if the temperature rises 2-4 degrees, turn an LED on
  else if (temperature >= baselineTemp + 2 && temperature < baselineTemp + 4) {
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  } // if the temperature rises 4-6 degrees, turn a second LED on
  else if (temperature >= baselineTemp + 4 && temperature < baselineTemp + 6) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
  } // if the temperature rises more than 6 degrees, turn all LEDs on
  else if (temperature >= baselineTemp + 6) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  }
  delay(1);
}
```

Constants are similar to variables in that they allow you to uniquely name things in the program, but unlike variables they cannot change. Name the analog input for easy reference, and create another named constant to hold the baseline temperature.
For every 2 degrees above this baseline, an LED will turn on.
The temperature is being stored as a float, or floating-point number. In the setup you're going to use a new command, **Serial.begin().**
This opens up a connection between the Arduino and the computer, so you can see the values from the analog input on your computer screen.
The argument 9600 is the speed at which the Arduino will communicate, 9600 bits per second. You will use the Arduino IDE's serial monitor to view the information you choose to send from your microcontroller. When you open the IDE's serial monitor verify that the baud rate is 9600.
Next up is a **for() loop** to set some pins as outputs. These are the pins that you attached LEDs to earlier. Instead of giving them unique names and typing out the pinMode() function for each one, you can use a for() loop to go through them all quickly.
In the loop(), you'll use a local variable named **sensorVal** to store the reading from your sensor. To get the value from the sensor, you call **analogRead()** that takes one argument: what pin it should take a voltage reading on. The value, which is between 0 and 1023, is a representation of the voltage on the pin.
The function **Serial.print()** sends information from the Arduino to a connected computer. You can see this information in your serial monitor. If you give Serial.print() an argument in quotation marks, it will print out the text you typed. If you give it a variable as an argument, it will print out the value of that variable.

With a little math, it's possible to figure out what the real voltage on the pin is. The voltage will be a value between 0 and 5 volts, and it will have a fractional part (for example, it might be 2.5 volts), so you'll need to store it inside a float. Create a variable named voltage to hold this number. Divide sensorVal by 1024.0 and multiply by 5.0. The new number represents the voltage on the pin.
Just like with the sensor value, you'll print this out to the serial monitor.
The datasheet for this sensor explains that every 10 millivolts of change from the sensor is equivalent to
a temperature change of 1 degree Celsius. It also indicates that the sensor can read temperatures below 0 degrees. Because of this, you'll need to create an offset for values below freezing (0 degrees). If you take the voltage, subtract 0.5, and multiply by 100, you get the accurate temperature in degrees Celsius. Store this new number in a floating point variable called temperature.
Now that you have the real temperature, print that out to the serial monitor too. Since the temperature variable is the last thing you're going to be printing out in this loop, you're going to use **Serial.println().** This command will create a new line in the serial monitor after it sends the value.

With the real temperature, you can set up an **if()...else** statement to light the LEDs. Using the baseline temperature as a starting point, you'll turn on one LED on for every 2 degrees of

temperature increase above that baseline. You're going to be looking for a range of values as you move through the temperature scale.

The **&&** operator means "and", in a logical sense. You can check for multiple conditions: "if the temperature is 2 degrees greater than the baseline, and it is less than 4 degrees above the baseline."

If the temperature is between two and four degrees above the baseline, the corresponding block of code turns on the LED on pin 3 as well.

The Analog-to-Digital Converter can only read so fast, so you should put a small delay at the very end of your loop(). If you read from it too frequently, your values will appear erratic.