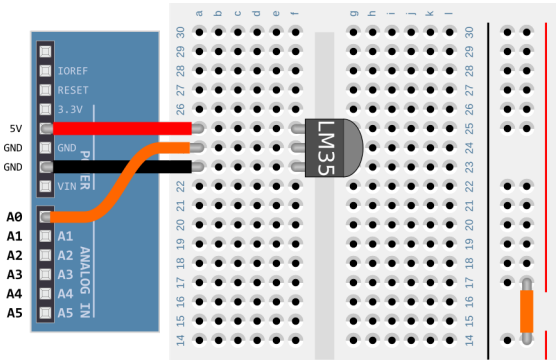
# Microcontrollers Systems and Interfacing

**week 5 experimental lab**

## 1 Measure a temperature




- Find the LM35 temperature sensor in your hardware kit. (The number is printed on the device.)

  Note: there are also two BC548 transistors in your kit which are similar to the LM35. Check very carefully that you have found the LM35 and not one of the transistors.

  **Do not** connect the LM35 to the breadboard! First, prepare your breadboard for the LM35. Connect the following to three consecutive rows, in this order:
  - 5 V
  - A0
  - GND

  When the connections are made, prepare the software.

- Write a program that prints the value of A0.

  This will be just like the programs you have already written to display the value of A0.

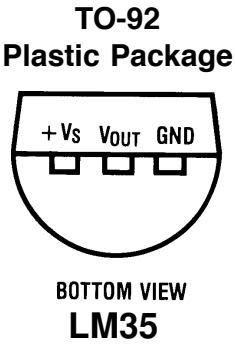
```
void setup() {
  Serial.begin(115200);
}

void loop() {
  int a0 = analogRead(A0);
  Serial.println(a0);
}
```

- Make sure the program is working.

  Run the program and open the serial monitor. Check that it is printing more-or-less random numbers.

- Find the 5 V and GND pins on the LM35.

  Study the picture on the right. It shows the pins on the LM35 from *below*, with the pins pointing towards your eyes. With the flat side of the device pointing up, the left pin is the $+V_s$ (5 V) connection and the right pin is GND. (The central pin is the output, no matter how you hold the device.)




- Connect the LM35 to the circuit.

  Make sure the 5 V pin connects to the 5 V row on the breadboard. For example, if you connected the wires as shown in the diagram above (5 V at the top and GND at the bottom) then the flat side of the device must face towards the left.

  As soon as the device is plugged in, check the output from your program. The serial monitor should be showing a value in the range 30–60. If the value is outside this range then **unplug the device immediately** and check your wiring.

  Keep holding the plastic body of the device for at least 30 seconds. If it becomes hot then **unplug the device immediately** and check your wiring. (You have probably connected it backwards. If you do not disconnect it immediately, the device will become so hot that it is permanently destroyed.)

  You can check that the device is working by warming it between your fingers. The voltage from the device should rise as it is warmed. If you let go, and blow on it to cool it, the voltage should decrease again.

### 1.1 Convert the value read from A0 into a temperature.

Refer to the diagram in the data sheet (shown on the right). The output from the device is '0 mV + 10 mV/ °C'.

The value read from A0 has a range of 0 to 1023. It will be 0 when the LM35 output is 0 mV and 1023 when the LM35 output is 5000 mV (5 V), which would represent a temperature range of 0 °C to 500 °C.

The conversion from `analogueRead` value to a temperature is therefore a conversion from a value in the range 0–1023 to a value in the range 0–500.

The Arduino library has a useful function called `map` that performs this kind of conversion:

## Typical Applications

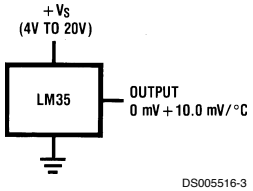



**FIGURE 1. Basic Centigrade Temperature Sensor (+2˚C to +150˚C)**

```
output = map(input, inLow, inHigh, outLow, outHigh);
```

The function takes five parameters that specify an *input* value, the expected range of input values `inLow` to `inHigh`, and the desired range of output values `outLow` to `outHigh`.

```
int temp = map(a0, 0, 1023, 0, 500); // convert A0 value 0..1023 to temperature 0..500
```

- Convert the raw A0 values into centigrade degrees.

  Create a new `int` variable (called `temp`, for example) and use the `map` function to initialise it with the temperature represented by `a0`. Print the temperature instead of the raw value. Make sure it looks reasonable based on the temperature of the room and your body temperature when you hold the plastic body of the LM35.

- Improve the precision of your conversion.

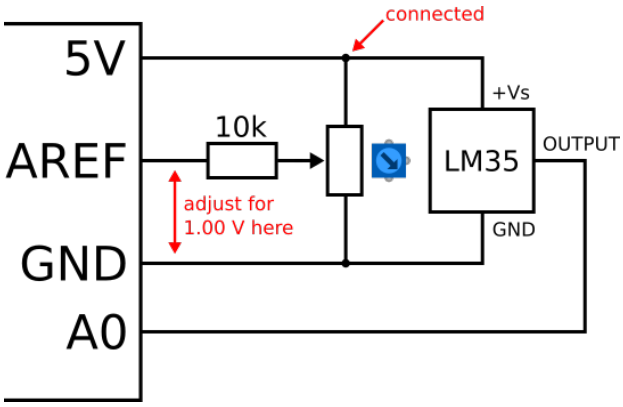
  The `map()` function uses *integer* arithmetic. This is reducing the resolution of your temperature sensor. Write your own version of `map()` that uses floating-point arithmetic to achieve the best possible resolution.

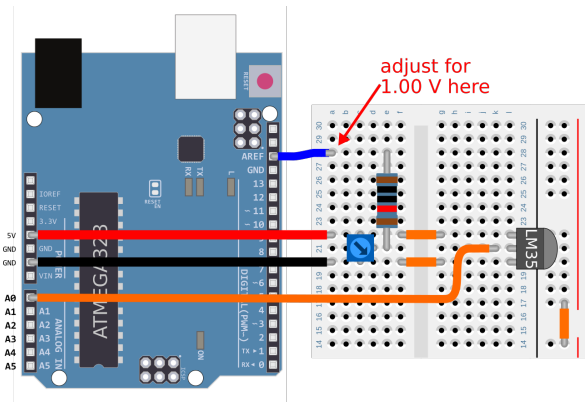
# 2 Challenges

## 2.1 Increase the resolution of your temperature sensor

The microcontroller's analogue-to-digital converter normally converts an input voltage range of 0 V–5 V to a digital range of `0–1023`.

The input voltage range can be changed by applying a reference voltage $V_{REF}$ to the AREF pin (next to digital pin 13) and then calling `analogReference(EXTERNAL)` in your `setup()` function. The microcontroller's analogue-to-digital converter will then convert an input voltage range of 0 V–$V_{REF}$ to a digital range of `0–1023`.




Use a potentiometer to create a variable analogue voltage. Connect this to one terminal of a 10 kΩ resistor, and the other terminal of the resistor to the AREF pin (*not* IOREF!) of the microcontroller. (The resistor is important.) Set your multimeter to its 20 V range. (From 'off', turn the selector knob three positions anti-clockwise.) Measure the voltage between the AREF side of the resistor and GND. Set the potentiometer so that this voltage is exactly 1.00 V. (Don't forget to turn your multimeter off.)

Modify your code to use the new range of input values, which will be `0` for 0 V, and `1023` for 1 V (100 °C). Your temperature resolution should now be approximately 0.1 °C.

# Microcontrollers Systems and Interfacing

**week 3 reference material**

## A Scale conversion: mapping between different value ranges

The `map(v, a, b, x, y)` function converts a value $v$ from the range $[a, b]$ to the range $[x, y]$. Mathematically:

$$\text{map}(v, a, b, x, y) = (v - a) \times (y - x) \div (b - a) + x$$

For example, to map from old Farenheit degrees in the range $[32, 212]$ to metric centigrade degrees in the range $[0, 100]$, the relationship is

$$\begin{aligned} \text{map}(v, 32, 212, 0, 100) &= (v - 32) \times (100 - 0) \div (212 - 32) + 0 \\ &= (v - 32) \times 100 \div 180 \\ &= (v - 32) \times 5 \div 9 \end{aligned}$$

(which is the formula we recognise and remember).

Beware that the built-in `map()` function uses integer arithmetic. If you want to preserve as much accuracy as possible, you should write your own version that uses floating point arithmetic.