Exercise 9: Temperature sensor

Equipment

For this exercise you will need:

- 1 x Arduino Uno
- 1 x TMP 36 GZ *or* LM 35 DZ
- Wires

Messy code? Use **ctrl+T**

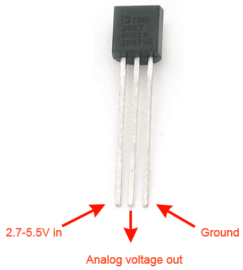


Figure 1: **Pin Layout:** the flat side of the "head" is turned towards you

Setup

The microcontroller has a circuit inside called an analog-to-digital converter (ADC), you saw in the last exercise how voltage is represented as a 10-bit binary value.

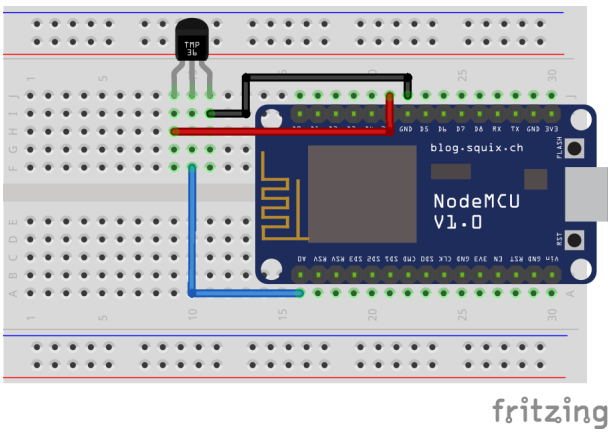
- The pin layout of the two sensors is the same. See fig. 1
- Connect the output of the sensor to the analog input (A0).

Questions & Exercises

9a: Look at the datasheet for your specific sensor (google it). What is the conversion from the A0 input to voltage? Formulate it as a singular expression.
9b: What would the output of this code be?

```
char c;
for (int i = 0; i < 4; i++) {
    c = '0' + i*2;
    Serial.print(c);
    Serial.write(176);
}
```

9c: What is the difference between print() and write()
9d: Read the voltage on A0.
9e: Convert the voltage into $°C$ and print it to the serial monitor. (You have to print the ° character, very important! Temperatures are not measured in Coulombs. Maybe this will help)
9f: Try heating and cooling the sensor. Is it working?



Hint

The conversion will depend on a couple factors; do you need to account for negative values? What is the max voltage input? What is the analog value associated with said input? What is the sensitivity of the sensor?

Exercise 10: Make diodes light up depending on temperature

Equipment

For this exercise you will need:

- 1 x Arduino Uno
- 1 x TMP 36 GZ *or* LM 35 DZ
- 1 x Red LED
- 1 x Green LED
- 1 x Yellow LED
- 3 x $\sim 60 - 220\Omega$ Resistors
- Wires

Setup

- Connect the LEDs as in exercise 2
- Connect the TMP36GZ/LM35DZ as in exercise 9

Questions & Exercises

10a: Measure the temperature changes that you are able to cause by heating the sensor with your fingers and cooling it by blowing air on it.
10b: Make the diodes light up depending on the measured temperature. Use these values you measured before to calibrate your sensor.

- Red = High
- Yellow = Medium
- Green = Low

Exercise 11: Output temperature to LCD-screen

Equipment

For this exercise you will need:

- 1 x Arduino Uno
- 1 x TMP36GZ *or* LM35DZ
- 1 x LCD-screen with I2C module
- Wires

Messy code? Use **ctrl+T**

Reading

Chapter 13

Setup

- Install the LiquidCrystal_I2C library by Frank de Brabander from within the Arduino IDE and include it in your sketch

```
#include <LiquidCrystal_I2C.h>
```

- Connect the temperature sensor.
- Connect the four pins on the LCD to the Uno.

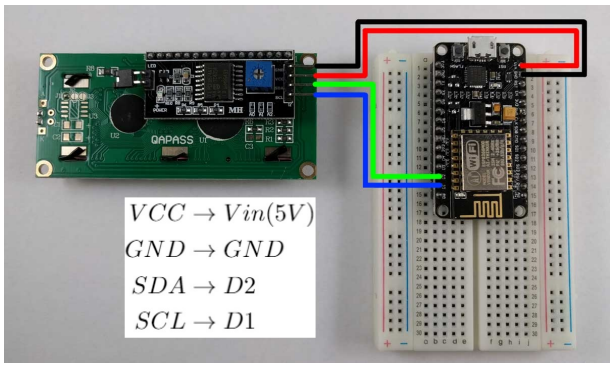


Figure 2: **I2C connection**

Questions & Exercises

11a: What is I2C?
11b: How can you save computational power when printing on the LCD?
11c: Write a program which:

- Reads the temperature using the temperature sensor.
- Print the temperature to the LCD.
- Show some warning when the sensor gets too hot (e.g. *!!!*).
- *Optional* Make at least ten measurements of the temperature between each time the display is updated. Have the temperature be an average of your measurements. This should make the shown temperature more stable.

Hint

If you have trouble finding the address of the LCD, you can use this code to find it (Usually the address is 0x27). You can use the code to test if the Arduino can "see" the LCD.

Exercise 12: Catch the LED game

Equipment

For this exercise you will need:

- 1 x Arduino Uno
- 5 x LEDs
- 5 x $\sim 60 - 220\Omega$ Resistors
- 1 x button
- Wires
- LCD Screen

The ternary operator:
`condition ? IfTrue : IfFalse`
Ex.
`(a==5) ? "a is five" : "a is not five"`

Setup

- Setup the LEDs close together in one straight line on the breadboard.
- Connect the five LEDs to five different digital pins.
- Connect the button.

Questions & Exercises

The five LEDs should flash in a consecutively pattern, one after the other. The goal of the game is to push the button when the middle LED light up. **12a:** Make the LEDs flash.
12b: When the button is pushed, print out the number of the lit LED to the LCD. Is the timing right?
12c: Implement a specific flashing pattern when the player pushes the button at the right time. If the player missses the LEDs must turn off for 1 second.
12d: Also, count the number of hits and misses. Print out these counters from time to time.
12e: Make the game progressively harder, by increasing the flashing speed every time the player hits.
12f: When the hit counter equals 8 the LEDs must flash randomly, one at a time. At 10 the player wins.
12g: Setup Doxygen and document your code. The hand-in for this exercise should include a fully-made Doxygen.

Hint

In this exercise you have to deal with timing. You probaly want different parts of your program to run at different frequencies e.g. maybe you shouldn't update the LCD everytime you check if the button is pushed. There are many ways of dealing with timing, the easiest is a counter as seen here

```
int count = 0;
loop(){
    if(count%2 == 0){
        // this happens every 20ms
    }
    if(count%4 == 0){
        // this happens every 40ms
    }
    if(count%100 == 0){
        // this happens every second
    }
    count++;
    delay(10); //wait 10ms
}
```