Additional information for the teaching unit ‚SMB–Science Magic Box‘

Content

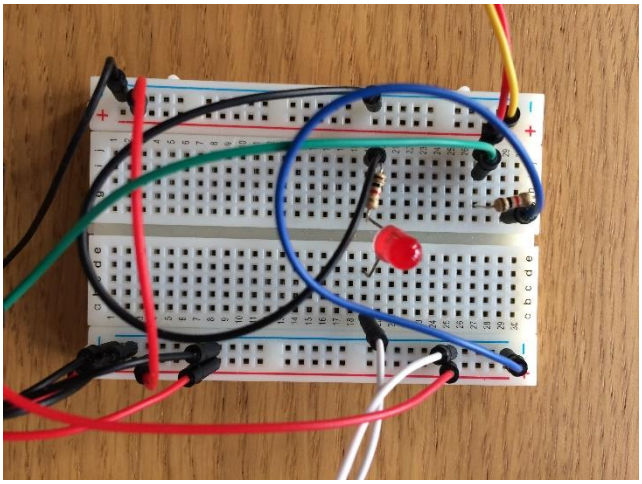
Böhm-Jacopini’s Theorem 1
Breadboard 1
Linearity 2
Calibration 2
Using PLX-DAQ data logger 3
Showing that your Arduino is alive–Blink on-board LED 3

Böhm-Jacopini’s Theorem

Even though for an experienced programmer it could seem rather obvious, from a logical-mathematics point this theorem is not trivial. It states that any sequential procedural program (a program in which instructions are processed by the microcontroller one after the other, like your C++ program) can be written with instruction based on procedures (subprograms) calls, branches (IF…THEN…ELSE… instructions) and loops. It is important because all the high-level programming languages offer a set of instructions based on this important result.

Breadboard

A breadboard is a circuit-builder device, which allows you to connect devices, sensors and microcontrollers using wires, trying circuit solutions that are not permanent, so that you can proceed with trial and error strategy, and test your projects and ideas. They are normally solderless, meaning that you don’t need to solder anything on them. The wires just must be plugged into the holes. The power and the GND signals are provided by the microcontroller, connecting wires to + and – holes columns. The other connections must be done following the sensors datasheets. The breadboard circuits are not meant to be permanent. When you see that all is working properly for your project, you make the final circuit with soldering, and the breadboard will be used to test the next project.



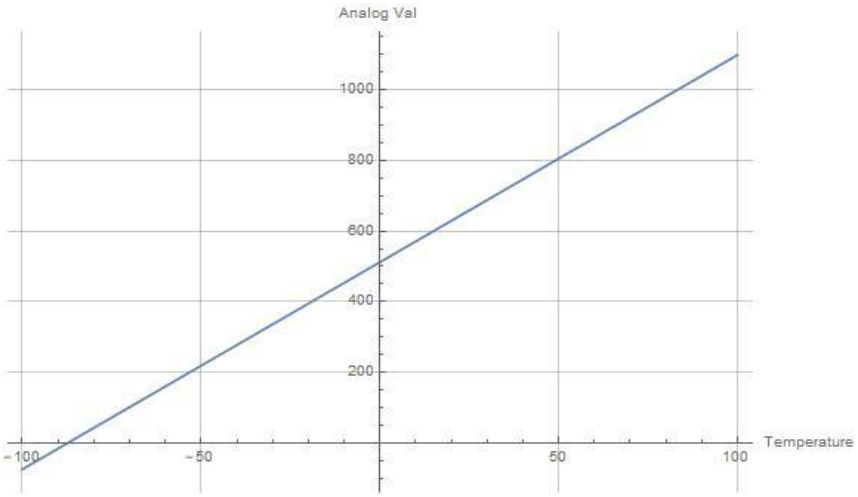
Linearity

The ideal transformation from one physical quantity (signal) into another would happen with a linear dependence between the two, which can be shown in the figure below, where the dependence of one quantity on the y-axis from the other on the x-axis is represented by a straight line. Normally the dependence is not perfectly linear in all the covered range of values the sensor can measure, but the closer the dependence is to a straight line, the easier will be the conversion and the better the accuracy will be. Which is why usually the sensor is considered to work with good approximation results in its linearity range.

An acquired physical quantity is converted into another following these steps:

a) The physical signal (light, sound, force, energy …) is acquired by the sensor and converted into an electrical signal.
b) The electrical signal is transformed into a number available to the processor.
c) The number is processed and transformed by the processor into another number, and then used to do an action with an actuator transducer.
d) The actuators convert the number into electrical signals that are ready to be output.
e) The electrical signal is finally transformed into a physical signal (e.g. sound, light).

In step c) there is also a conversion from one number to another, but as this transformation is entirely digital and completely processor-controlled, the coding and the math operation needed for the conversion guarantee that this transformation is linear. (If linearity is needed; sometimes this transformation will not be linear because the two signals don't have a linear dependence.)

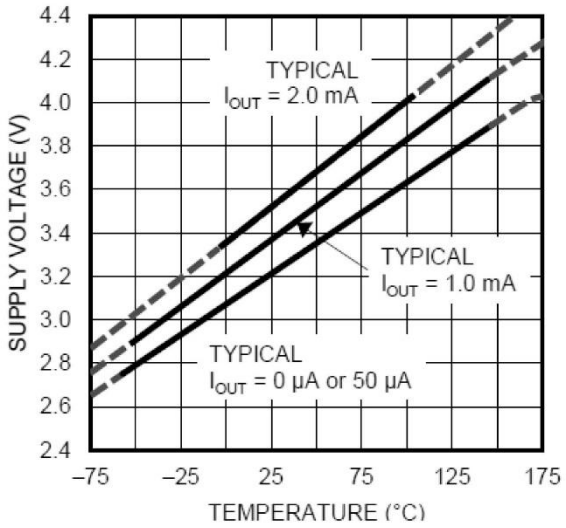


Calibration

A sensor usually needs calibration before being considered reliable in its measurements.
The calibration usually is performed knowing at least two signal levels obtained through another pre-calibrated sensor or another measuring tool (like a good working thermometer for temperature), and adjusting the sensor's digital values according to a linear transformation that matches the two known values to the two corresponding numerical values output by the sensor.
For instance the transformation needed for the LM35 temperature sensor is:

```
val=analogRead(A0);    //Connect LM35 on Analog 0 input
temperature = (double) val * (3.3/10.24);  // Vref on "Arduino Due" is 3.3 V
```

This linear transformation converts the analogue value read from Analog Input A0 first as a Voltage with maximum scale 3.3, and then as a number expressed by the ten bits of Analog inputs that allow an electrical level to be represented by numbers from 0 to 1023.
A transformation like this is called a Transfer Function, because it transfers the numerical meaning from a physical quantity to another, or to an electric level. Usually devices producers or resellers provide transfer functions.



Transfer function for TI - LM35 Temperature sensor

Using PLX-DAQ data logger

The Arduino Integrated Environment offers a datalogger, called PLX-DAQ, which allows the programs running on Arduino to collect data, format them in Spreadsheets compatible form, and to graph them versus time or export them in a spreadsheet. The advantage of this is that the collected data can be stored, examined and analysed with different statistical and analytical methodologies. You can download PLX-DAQ software for free from the Arduino Website.

Showing that your Arduino is alive–Blink on-board LED

Usually an automation program works forever, or, at least, as long as your microcontroller has power. For this reason, a strategy sometimes called “WatchDog” (technically WD) is used to show externally that your program is doing what it is meant to do. The simple thing to do is to insert in the code three lines, one that turns on an LED (normally the Arduino on-board LED), one that tells Arduino to wait a few milliseconds after processing the code, to allow your eyes to see the LED light, and the third that turns off the LED. What you see externally is the on-board LED blinking, telling you that the main loop is executed correctly. Should the code for some reason (usually an unwanted bug caused by a wrong coding logic) exit the loop, the blinking would cease, telling you that the microcontroller is not working properly.