Report: Temperature measuring device

Loana Grom

winter term 2022/2023, Universität Hamburg

1 Overview

A temperature measuring device was built as part of the one-week block course "Practical Measurement Electronics and Interfaces in Ocean Sciences" by Niels Fuchs and Markus Ritschel at Universität Hamburg. During the course the theory of technical measuring instruments was covered and applied by each participant creating and programming a temperature measuring device.
The main components of the built temperature measuring instrument are an Arduino UNO duplicate and a DS18B20 digital thermometer. 12-bit temperature measurements of the DS18B20 can be recorded in degrees Celsius and read out from the temperature measurement device at specific time intervals. For the temperature device I built, the LG device, I have chosen a configuration where the temperature is read out every 10 seconds with a spot and an average measurement. This report serves as a data sheet for the built temperature measuring device.

2 Components

To set up the temperature measuring device, an Arduino UNO duplicate with a data-logger shield (hereafter referred to as Arduino) is connected to the DS18B20 temperature sensor.

2.1 Arduino UNO and data-logger shield

The Arduino UNO is a microcontroller board with a 16 MHz processor, 14 digital input/output pins, 6 analogue inputs, an external power supply, an USB port and a reset button. It can be either powered by the external power supply or via the USB connection. Additional components can be attached via so-called shields, that are stacked onto the Arduino board. The Arduino is equipped with an analogue-to-digital converter (ADC), which can process up to 5 V and outputs a 10-bit number. The maximum voltage resolution is therefore 0.0005 V. In order to read out the temperature data of the temperature sensor with the Arduino, a so-called "sketch", i.e. the Arduino programming code, must be written and uploaded to the Arduino. Sketches are written and compiled in the Arduino IDE. The Arduino specifications are taken from its data sheet [Arduino, 2021].
For setting up the temperature measuring device in the course, the power supply via the USB connection (5 V) and a baud rate of 9600 was selected. A data logger shield with an SD card port and an RTC powered by a battery was connected to the Arduino board. The pins used and the code written are explained in section 3.

2.2 DS18B20 digital thermometer

The DS18B20 digital thermometer delivers 12-bit temperature measurements in degrees Celsius with an accuracy of $\pm 0.5\,^{\circ}\mathrm{C}$ between temperatures of $-10\,^{\circ}\mathrm{C}$ and $85\,^{\circ}\mathrm{C}$. Bus communication is initiated by the 1-Wire bus control with only one control signal. Over the data line all data and commands are transmitted from the least significant bit (LSB) to the most significant bit (MSB). The thermometer can be operated with a supply voltage V_{DD} between 3 V and $5,5$ V and it measures temperatures in the range of $-55\,^{\circ}\mathrm{C}$ and $125\,^{\circ}\mathrm{C}$. Each sensor has an individual 64-bit serial code, which can be used to differentiate between

different sensors of the same type. The DS10B20 has a 3 pin configuration with a ground (pin 1: GND), a data input/output (pin 2: DQ) and a supply voltage pin (pin 3: V_{DD}). It consists of a 64-bit ROM, a scratchpad and the temperature sensor. Each component has its own main functionality, described in the following.
The 64-bit ROM stores the unique serial code of the unit. The bits are organised as follows: The least significant 8 bits contain the device's family code, followed by the next 48 bits that store the serial code. The most significant 8 bits of the ROM code contain a cyclic redundancy check.
The scratchpad of the DS18B20 is the memory of the unit. The main functionality for the temperature measuring device built in the block course is the storage of the digital output of the temperature sensor in its 2-byte temperature register. Byte 0 of this temperature register contains the LSB and byte 1 the MSB of the temperature sensor output. Both bytes are read-only and are transmitted during read-out via the 1-Wire bus starting with the LSB of byte 0.
The direct-to-digital temperature sensor stores temperature measurements as a 16-bit sign-extended two's complement number in byte 0 and byte 1 of the scratchpad's temperature register. Bit 0 to bit 3 of this number represent the decimal places, bit 4 to bit 10 the digits before the decimal point and bit 11 to bit 15 the sign (0: positive sign, 1: negative sign). The DS18B20 specifications are taken from its data sheet [Maxim-Integrated, 2019].

3 Technical Description

3.1 Wiring

The Arduino and the DS18B20 digital thermometer are connected via three cables and a resistor as follows (see fig. 1): The ground pins of the thermometer and Arduino are connected. The data input/output and the supply voltage pin of the DS18B20 are connected to the digital pin 4 and the 5V output supply pin of the board, respectively. A pull-up resistor of $4,7\,k\Omega$ is interposed between the latter two connections. Since the 1-Wire bus can only transmit High or Low states, a switch is used to switch between states and connect the circuit to ground. When the switch is open, the temperature sensor is supplied with voltage, thus a temperature can be measured. For a closed switch, the voltage is conducted to ground, therefore the unit is in Low state. To prevent a direct connection between ground and V_{DD}, i.e. a short circuit, the pull-up resistor is needed.

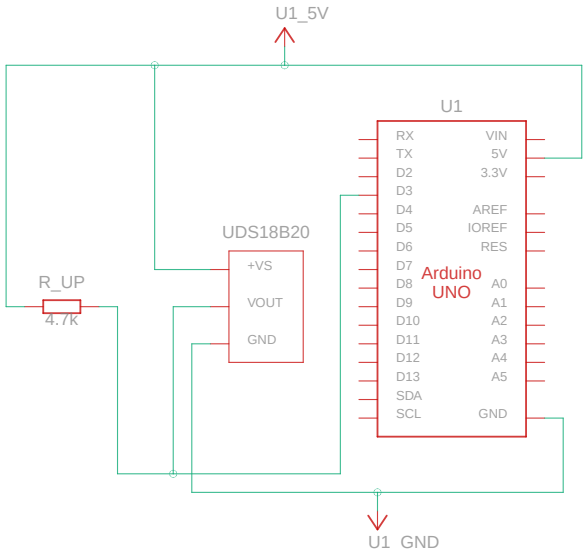

Figure 1: Circuit diagram. R_UP: resistor ($4.7\,k\Omega$), U1_5V: voltage ($5\,V$), U1_GND: ground, UDS18B20: DS18B20 sensor and U1: Arduino.

3.2 Arduino code

The Arduino programming code, the sketch, is written in the C++ language and is used to send commands to and read information from the connected sensor. We used the sketch to initiate temperature measurements and conversions, read the readings and interpret them from bits into a temperature value in degrees Celsius and store them on the SD card.
The sketch is divided into three main components. The first part is used to load libraries and define constants and variables. The second component is the setup function. The setup function is called once at power up of the Arduino. The third component is the void loop, which is called continuously, when the Arduino is powered.

3.2.1 Libraries and helper functions

To operate the DS18B20 digital thermometer with the Arduino, the SD, SPI, RTClib and OneWire libraries must be loaded at the beginning of the sketch. The SD library enables temperature data to be written to the SD card. The SPI (Serial Peripheral Interface) library is used for fast communication with the DS18B20 sensor. The Real Time Clock (RTC) can be initialised and set with the RTClib library. Finally, the OneWire library is used to access the DS18B20 digital 1-Wire thermometer.
The helper functions helper.ino by Markus Ritschel and ds18b20.h are also included into the sketch. The helper.ino function can be used for timestamp creation, string formatting to two digits and simultaneous writing to SD card and terminal output. The ds18b20.h holds all required HEX commands for working with the DS18B20 sensor that are used in the void loop (see sec. 3.2.3).

3.2.2 Void setup

In the void setup, the `Serial.begin()` command is first used to set the baud rate for serial communication to 9600. Then the RTC and SD modules are initialised and checked to see if they are running. If the modules are not running, an error message is displayed on the serial monitor with the `Serial.println()` command. If the RTC clock is not running, the RTC is set to the date and time of the sketch compilation. Finally, the header is output to the terminal and the SD card using the `printOutputln` command of the helper.ino function.

3.2.3 Void loop

In the void loop, transaction sequences for accessing the temperature sensor are initialised. According to the sensor specification, they must have the following sequence: 1. initialisation, 2. ROM command and 3. DS18B20 function command. The initialisation sequence is the first command to start communication with the sensor. It consists of a reset pulse sent from the bus master to the sensor and a presence pulse sent from the sensor to the bus master. The presence pulse is the sign for the bus master that the sensor is ready for operation. The Rom commands are used after initialisation to perform operations on the ROM. The DS18B20 function commands, are used to write and read scratchpad memory and initiate temperature conversion.
The void loop is divided into three different transaction sequences. All sequences start with the `ow.reset()` command, the initialisation part of the sequence. Part 2 and 3 (ROM and DS18B20 function commands) of the sequences start with the commands `ow.write()`.
The first sequence in the void loop reads the ROM code. It uses the command `Read ROM [33h]`, which reads the 64-bit ROM code of the DS18B20 sensor. The 8 bytes of this number are then stored in the variable `rom_code` and it is checked whether the connected sensor is of type DS18B20. The unique serial number is then stored in the string `registration_number`.
The second sequence converts temperature measurements. Since no information from ROM is needed in this sequence, the command `Skip Rom [CCh]` is used, which addresses the sensor on the bus without sending ROM code information. Then the command `Convert T [44h]` is used. This command triggers a single temperature conversion and stores the resulting temperature data in the scratchpad memory.
The last sequence starts with the `Skip Rom [CCh]` command again, followed by a `Read Scratchpad [BEh]` command, that is used to read the temperature data of the scratchpad starting with the LSB of byte 0. The 9 bytes of scratchpad data are stored in the variable `sp_data`. Since the temperature information is stored in the first two bytes of the scratchpad, this data is shifted by 8 bits, converted into two-digit temperature information and stored in the variable `tempCelsius`. In the next step, the data of the time stamp, the sensor ID and the spot and average temperature measurements are stored and written to the terminal and the SD card with the commands `printOutput` and `printOutln`. The temperature measurements are taken every second with a delay of the difference between the current milliseconds and the next even 1000 milliseconds. Spot and average measurements are stored every 10^{th} second, with the average being the average temperature measurement of the last 10 seconds.

4 Measurements

To calibrate the temperature measuring instruments and calculate the time constant, temperature measurements were taken in a small water tank in the ice laboratory. A Greisinger thermometer (which one??) served as the reference sensor for the calibration. The sensors of the self-made temperature measuring devices were fixed around the temperature sensor of the Greisinger thermometer. This made it possible to have all the sensors in approximately the same position, which is necessary for calibration and comparison between the sensors.
The Greisinger thermometer had to be read out individually, since it does not measure a temperature time series. Two time series, hereafter referred to as calibration and time constant time series, were noted. The sensors were put from air into water, creating a rapid change in temperature (time constant time series) and then remained in water with the water being slowly heated until 20 °C were reached (calibration time series).

4.1 Calibration

To start the evaluation of the calibration measurement, all sensors first had to be brought to the same time axis. For this purpose, the time at which all sensors were placed in the water was used (shock time). In the time series of the temperature measuring devices (see fig. 2), a sharp drop in temperature can be seen for all sensors. The shock time of each sensor was then determined from the temporal temperature differences between the neighbouring time points t_i and t_{i+1}. The temperatures remained approximately the same from milliseconds 200000 with variations of about 0.06 °C. The shock time was defined as the first temporal temperature difference greater than 0.1 °C for each individual sensor from milliseconds 200000. The offset of the shock times between the Greisinger thermometer and the other temperature sensors was calculated and removed from the time series.
A second order polynomial was used to parameterise the calibration. For this procedure, the differences between the calibration and temperature sensor time series were first calculated. Using the Python library `np.polyfit`, a polynomial fit between the calibration temperatures and the differences was performed as a three-point calibration (see fig. 2). The polynomial fit was of second order and in the form of:

$$y = ax^2 + bx + c \tag{1}$$

with y: temperature differences, x: temperatures calibration time series and a,b,c: polynomial fit parameters. The parameters of the polynomial fit are shown in the table 1. The temperature time series were then corrected with the calibration by subtracting the polynomial function. The calibration is valid in a temperature range between 3 °C and 20 °C, as the calibration time series was only taken for this range.

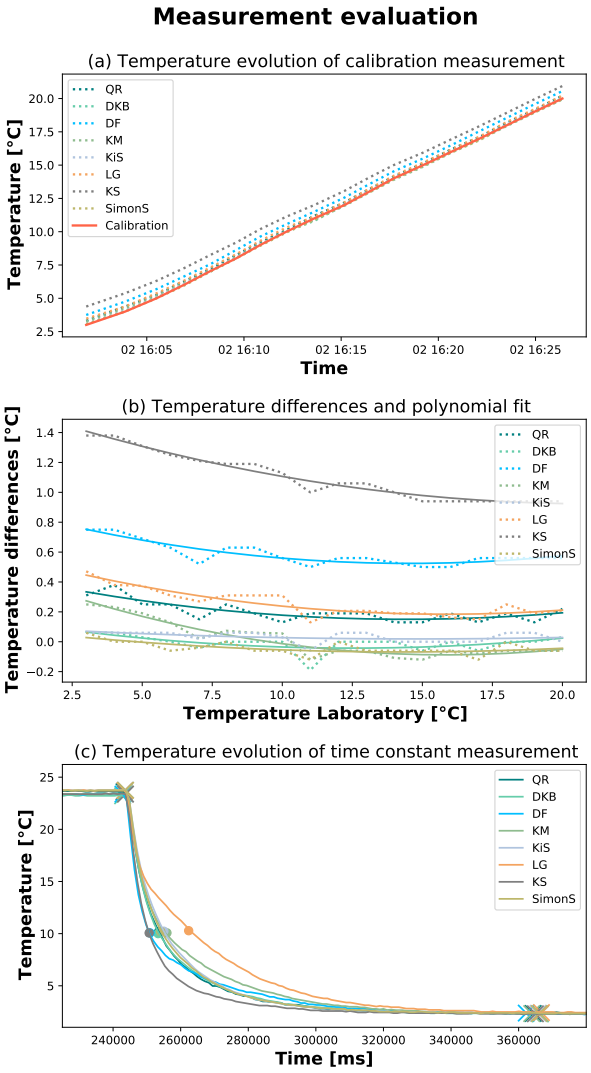

Figure 2: Temperature evolution of all sensors for calibation (a) and time constant time series (c). In (c): x: start and end times and dots: time constants. (b) shows temperature differences between sensors and calibration values and 2^{nd} order polynomial fit.

4.2 Time constant

The time constant of a temperature sensor is defined as the time the sensor needs to adjust to a change in temperature. This is the time until the sensor outputs 63.2% of the new changed temperature. This temperature and the time constant are calculated as follows:

$$T_\tau = T_{end} + 1/e * (T_{start} - T_{end}), \qquad \tau = t(T_\tau) - t_{start} \tag{2}$$

with T_τ: Temperature adjusted to 63.2% of changed temperature, T_{end}: changed temperature, T_{start}: start temperature, τ: time constant, $t(T_\tau)$ time when T_τ is reached, t_{start}: shock time.
Temperatures are defined as temperature averages over 30 seconds before start time (T_{start}) and after end time (T_{end}). The start time refers to the shock time and the end time is defined as the time, where the temporal temperature change over an interval of 100 seconds is first smaller than 0.1 °C. This provides the time from which the sensor has adapted to the new changed temperature.

4.3 Comparison of temperature measuring devices

The built temperature sensors differ slightly in time constant, calibration coefficients and measured temperatures (table 1). The largest difference between start temperatures is 0.5 °C between my own (LG) and the KiS sensor at 23.75 °C and the DKB sensor at 23.25 °C. The maximum difference in end temperature is 0.15 °C again between the LG sensor and the DKB sensor. Therefore it seems that the LG sensor measures higher temperatures and the DKB sensor measures lower temperatures compared to the other sensors in the temperature range between 2 °C and 24 °C. In the calibration evaluation, the highest differences of the sensor and Greisinger temperature measurements are found for the KS sensor with differences between 0.9 °C and 1.41 °C. The lowest differences are found for the SimonS sensor. My own sensor has the third highest differences.

	Temperatures [°C]				**Calibration coefficients**		
Sensor	T_{start}	T_{end}	T_τ	τ [s]	**a**	**b**	**c**
QR	23.65	2.38	10.20	253.49	0.0014	-0.0406	0.4431
DKB	23.25	2.29	10.00	253.33	0.0012	- 0.0298	0.1439
DF	23.38	2.33	10.07	250.92	0.0017	-0.0499	0.8866
KM	23.40	2.32	10.08	255.94	0.0022	-0.0695	0.4626
KiS	23.75	2.36	10.23	255.03	0.0004	-0.0109	0.0998
LG	23.75	2.44	10.28	262.37	0.0016	-0.0503	0.5825
KS	23.38	2.32	10.06	250.67	0.0015	-0.0624	1.5828
SimonS	23.74	2.35	10.22	254.09	0.0008	-0.0216	0.0861

Table 1: Temperatures of the measurements, time constant and calibration coefficients for every temperature measuring device.

Based on the comparison, it can be said that my own temperature sensor, the LG sensor, is not as sensitive to temperature changes as the other sensors. However, it keeps well to the actual measured temperature.

5 Specifications of the temperature measuring device

The built temperature measuring device, the LG device, has a temperature resolution of 0.0625 °C. This is the temperature resolution of the DS18B20 digital thermometer, since the digital value of the temperature is read by the Arduino. The temporal resolution of measurements is 10 seconds. The temperature sensor of the LG device can measure temperatures in a range of −55 °C to 125 °C. If the entire LG unit is placed in an environment to be measured, the temperature range must be limited to −40 °C to 85 °C as the Arduino board can only operate in this range. The device is calibrates in a temperature range of 3 °C to 20 °C. The accuracy of the measurements is ±0.5 °C in the temperature range of −10 °C to 85 °C.

References

Arduino. Arduino® uno r3 product reference manual. 2021. URL `https://docs.arduino.cc/static/4cfa87b75db487ab7d2fa4aef2aebf08/A000066-datasheet.pdf`.

Maxim-Integrated. Ds18b20 programmable resolution 1-wire digital thermometer. *Rev*, 6:20, 2019. URL `https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf`.