Practical Measurement Electronics and Interfaces in Ocean Sciences

Report

Konstantin Mrozik
konstantin.mrozik@studium.uni-hamburg.de

Abgabe: March 12, 2023

Contents

Goal

The aim of the experiment is to calibrate several temperature sensors with each other and to determine the time constant of the sensors. In addition, the measurements of the sensors are to be compared. In order to become familiar with the Arduino programming, a status LED is to be programmed that indicates the status of the measurement.

1 Error Theory and Calibration

To use a sensor in scientific experiments its very important to validate and calibrate it beforehand. There are 2 sources of error in scientific data acquisition, Systematic error and Random error. Random error is apparent in every measurement and we can not influence it, the only way to reduce random error is to use statistical averages which is not always possible. In our case we can reduce the random error by using a average measurement which integrates and averages all data points between two spot measurements. For Example: If we make a measurement every second we can either measure the temperature at the the specific second which is called a spot measurement, but if our sensor has a data aquisition rate of for example 10 Hz we can also add up the ten measurements in one second and then divide them by 10 to get an average measurement for each second. The other type of error is a systematic error which systematically influences all measurements of one sensor or experiment. Some Errors in the material or in the electronics can influence the data aquisition by systematically measuring a temperature 2 °C higher than the "real" temperature. Another type of systematic error is the difference in the internal clocks and therefore the difference in the timestamps. In our data aquisition we can influence the systematic error and therefore we minimize it as much as possible. To calibrate our sensors we use a known timestamp to set all timestamps of the different sensors. We also use a calibrated Thermometer to calibrate the temperature measurements of the different sensors. There are also different types of systematic errors. Either the error is a offset to the actual response which is constant for all measurements which is easy fixed by adding or subtracting a constant value to the data. Another type of error is the gain error which is dependent on the measurements and subtracts or adds a prcentage of the measured value, this error is fixed by deriving the percentage and removing/adding it to the data. Measurement can also experience a non linear error which is harder to remove, but if it is possible to derive the function of this error one can also add/subtract the error again.

2 Execution

3 Components

3.1 Arduino Uno

The Arduino Uno is a microcontroller board which can be progrmmed with C++ Code to do send and recive different electronic signals. The arduino board has different analog and digital inputs and outputs which can be connected to electrical cicuits to work with components like sensors, LEDs or diferent logical components.

3.2 LED

LED is short for Light Emitting Diode and therefore LED's are a special kind of Diode which emits light if a current is applied. Diodes are electrical components whose conductivity depends on the applied pooling. Classical semiconducter diodes are based on the physical properties of a pn-junction as explained in [1].

3.3 Temperature Sensor DS18B20

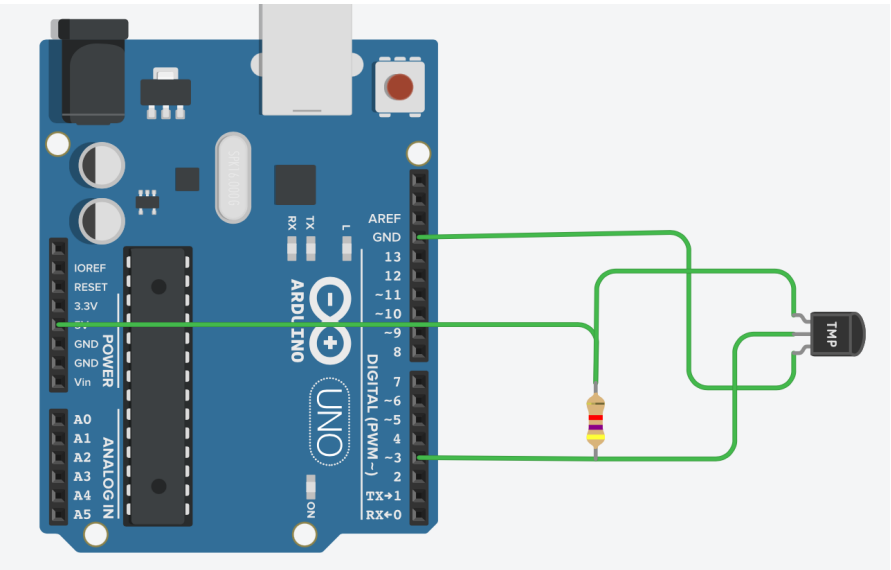
The DS18B20 is a programmable thermometer that can be controlled via a single cable. Special advantages of the temperature sensor are that it requires only one cable for control besides the ground connection and due to the unique 64-bit serial codes of the sensors it is possible to control multiple sensors with a single cable. In the range of $-10\,°C$ to $85\,°C$, the temperature sensor has an accuracy of $\pm 0.5°C$ and can be used in a temperature range from $-55\,°C$ to $125\,°C$. With the help of a prasitic power mode it is even possible to operate the temperature sensor with only 2 connections (GND and DQ) whereby the power supply is then also provided via the signal connection. In our case, however, we used 3 cables for operation. Because the temperature sensor is connected to a 3-state port for its input signal, a pull-up resistor is used on the control line.

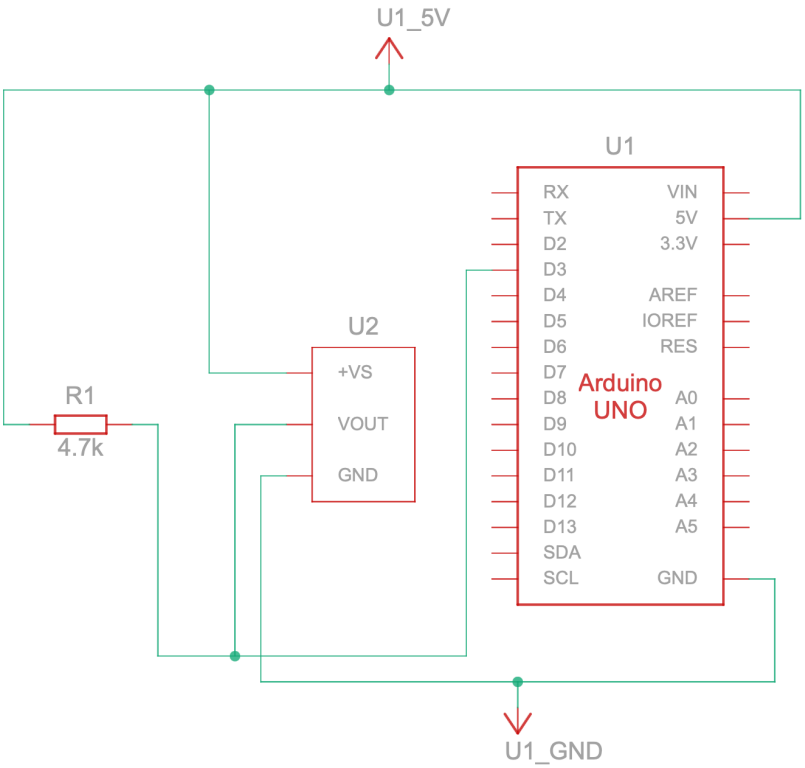
3.4 Experiment

The different Thermometers from the Arduino Circuits are connected to a handheld lab thermometer in a way that all thermometer tips align. For the expirement a watertank with a heating/cooling plate and a pump to mix up the watercolumn is brought to a temperature of $3\,°C$ while the surronding air temperature is at $25\,°C$. To calibrate the sensors, all connected sensors are immersed in the water bath at the same time. After a short time the temperature of the water bath is changed and increased to $20\,°C$. Each difference of $1\,K$ is noted with the corresponding timestamp.

3.5 Arduino Wiring

The arduino wiring is shown in Figures **??** and **??**.


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
TX
RX
L
ON
TMP


U1_5V
U1
RX
TX
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
SDA
SCL
VIN
5V
3.3V
AREF
IOREF
RES
A0
A1
A2
A3
A4
A5
GND
Arduino
UNO
U2
+VS
VOUT
GND
R1
4.7k
U1_GND

3.6 Arduino Coding

To setup the arduino first of all the void function setup() is programmed. In the setup the Serial output is initialized with a baud rate of 9600. If the RealTimeClock (rtc) is not running a error message is printed and the rtc will be started. For the first time running the code the internal clock is initialized with the local Computer time. Next up the SD card is checked and of its not found an error message is prompted. The different digital pins are put to the output mode to deliver information to the temperature sensor. At last the header for the data is saved in the SD card LOG file.
For the measurements a routine is defined in the function loop() for the arduino.
First two arrays are initialized for the scratchpad data and the rom code. Then the OneWire bus is reset and the Rrom code is read to save the registration number or ID of the sensor. Afterwards the OneWire test is reset again to save the temperature reading. The temperature data is converted from binary to decimal and then all readings are saved on the SD card. At the end a delay of 1000 milliseconds assures that the data is read every second.

3.7 Adjusting for Average meas

Before the void setup():

```
int cnt = 0;
float avg = 0;
```

In the void loop():

```
if (cnt < 9){
    avg = avg+tempCelsius;
  }
else{
    save_data_point(time_now,registration_number,tempCelsius,avg/10);
    cnt = 0;
  }
delay(1000);
```

In the save_data_point function:

```
void save_data_point(String time,String reg,float spot,float avg){
  printOutput(time);
  printOutput(", ");
  printOutput(String(millis()));
  printOutput(", ");
  printOutput(reg);
  printOutput(", ");
  printOutput(String(spot));
  printOutput(", ");
  printOutputln(avg);
}
```

3.8 Status LED

LED with different shades of blue,red and purple depending on the temperature.

```
void LED(int signal){
    if (signal < -55){
        digitalWrite(5,255);
        digitalWrite(6,0);
      }
      else if(signal > 125){
      digitalWrite(6,255);
        digitalWrite(5,0);
      }
      else if (signal > -55 && signal < 125){
        int brightness = ((signal + 55. )/180.)*255;
        analogWrite(5,255 - brightness);
        analogWrite(6,brightness);
      }
}
```

4 Evaluation

In figure 1 the raw data of the temperature sensors are plotted with their timestamp. The large time deviation of the sensor d39896f013c is particularly striking. In addition, a stepwise drop in temperature can be seen even before the water bath and it can be seen that the different sensors also show systematic differences in the water temperature. The slight noise in the measured values also shows that these are point values and not average values.
Figure 2 again shows the temperature data. A timestamp was added to the sensors that did not have a timestamp, according to the milliseconds recorded with the data. Also, using the df.diff() function of the panda package, the point at which the data drops off sharply (the point at which the sensors were placed in the water bath) was found and set to the same time point for all data. It can be seen that the data still has a systematic deviation in temperature and that each sensor has small differences in the time constant.

In figure 3 the temperature data of the displacement sensors were calibrated. The difference between the measured temperature of the sensors and the measured temperature of the reference thermometer after immersing the two sensors in the water bath was determined. The determined difference is then subtracted from the sensor values to calibrate them uniformly. You can see that the values of the sensors differ only slightly from each other.

By showing the calibration data with 0,5 °C added and remove with red lines in Figure 4 it can be seen that all values lay in the given accuracy of ± 0.5 °C.

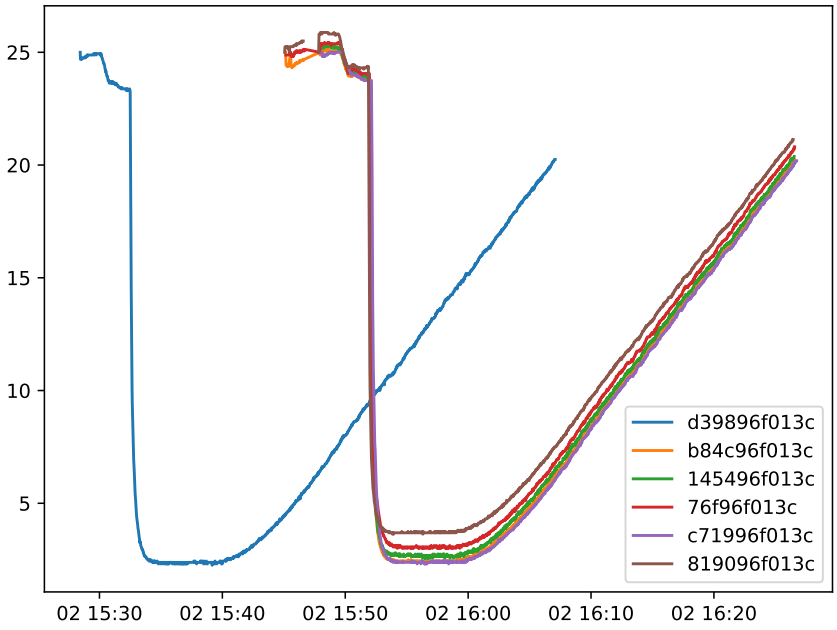


Figure 1: Raw data from all sensors with timestamps plotted. It can be easily seen that sensor d39896f013c largely derivates from the other sensors in time. The other sensors mostly are very close with their timstamp with very small variations in time.

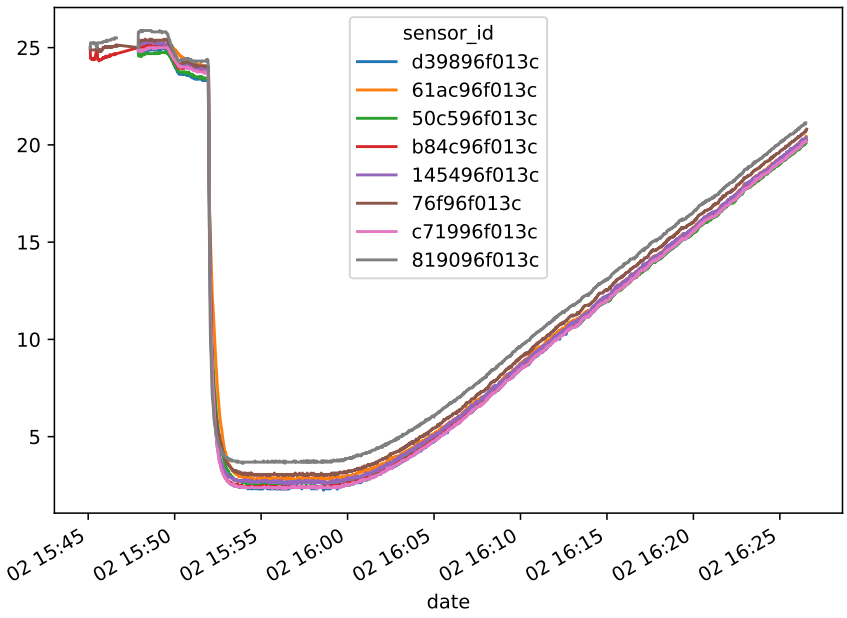


Figure 2: The timestamp adjusted data of all Sensors. Apparently the data still varies in Temperature and the different sensors have different time constants.

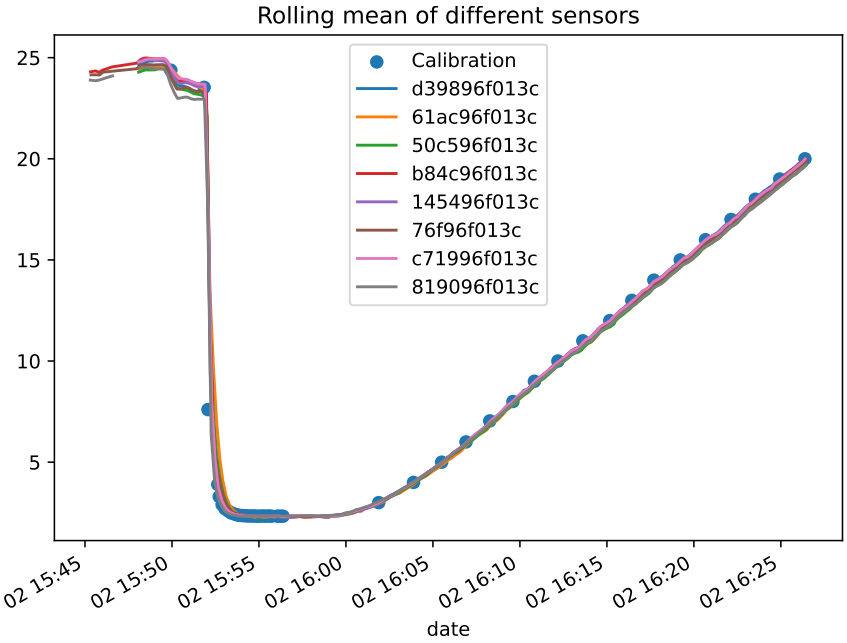


Figure 3: Rolling mean values of the temperature data and the temperature datapoints from the calibration measurement. The Values are adjusted with a calibration value at roughly 15:55 o'clock.

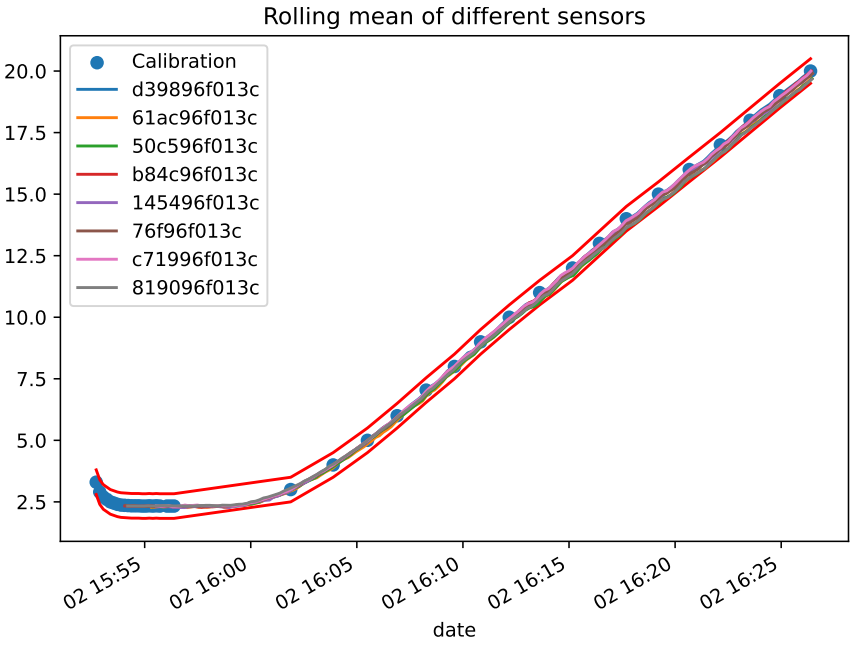


Figure 4: Just the rolling mean values while the thermometers are in the water bath. Withe red lines the deviation of 0.5 degree Celsius is shown.

To determine the time constant of the sensors, the value shortly before entering the water bath is compared with a value in the water bath and the difference is determined. Then a cut-off value is determined at which the value has changed by at least 63,2 %. Finally, the time difference between the start and the cut-off value is determined and the mean value of all sensors is calculated. According to the Gaussian error propagation, with an uncertainty of one second in the measured values, a value for the time constant of $(11{,}0 \pm 28{,}2)$ s results.

$$u_{time\_delta} = \sqrt{1^2 + (-1)^2)} = \sqrt{2} \tag{1}$$

$$u_{avg\_time} = \sqrt{(1/8 * \sqrt{2})^2} = \sqrt{1/32} = 4\sqrt{0.5} \approx 2.82843 \tag{2}$$

5 Conclusion

The strong deviation of the single sensor in time can probably be explained by a different time zone of the computer time (probably Turkey) of the respective PC. In addition, the gradual drop in temperature before inserting the sesnors into the water bath can be explained by the fact that the air in the container just above the water surface is probably colder than the air in the rest of the room.

Looking at the graphs of the temperature sensors, the readings are only slightly off, suggesting that the sensors are working properly. The calculated time constant seems to be legitimate compared to the literature values of 200 ms in water and 95 s in water. The value of $(11{,}0 \pm 28{,}2)$ s lies exactly between the two literature values, which is reasonable considering that it was a change of measurement in air and water.

References

[1] *Elektronik-Kompendium.* URL: https://www.elektronik-kompendium.de/sites/bau/0201113.htm.

[2] John D. Hunter. "Matplotlib: A 2D Graphics Environment". Version 1.4.3. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. URL: http://matplotlib.org/.

[3] Travis E. Oliphant. "NumPy: Python for Scientific Computing". Version 1.9.2. In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20. URL: http://www.numpy.org/.

6 Appendix