Reykjavík University

Mechatronics I

T-411-MECH

Laboratory 5

16/09/2020

Students:
Giacomo Menchi
Silja Björk Axelsdóttir

Teacher:
Joseph T. Foley

Contents

1 Main Tasks

As usual, the code for this assignment is available at the following Github repo:

`https://github.com/ru-engineering/lab-5-group-10.git`

1.1 Main Task 1

Task: Install an analogue light sensor to the Arduino Nano and print the results out to the Arduino Serial Monitor.

Pictures: The photos below show how we setup the circuit and one reading example.

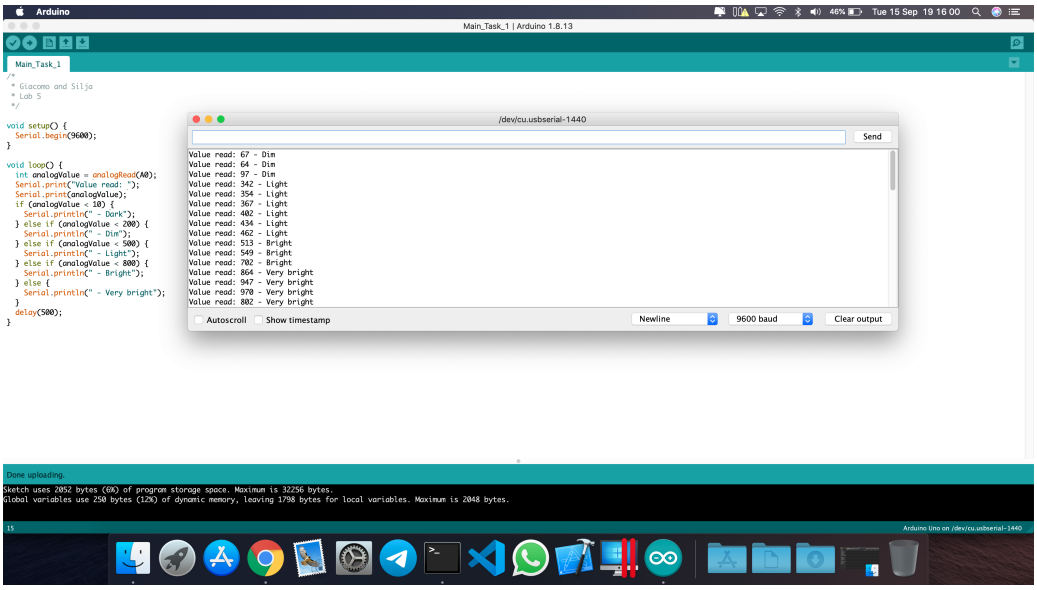

Figure 1: Light reading - console.

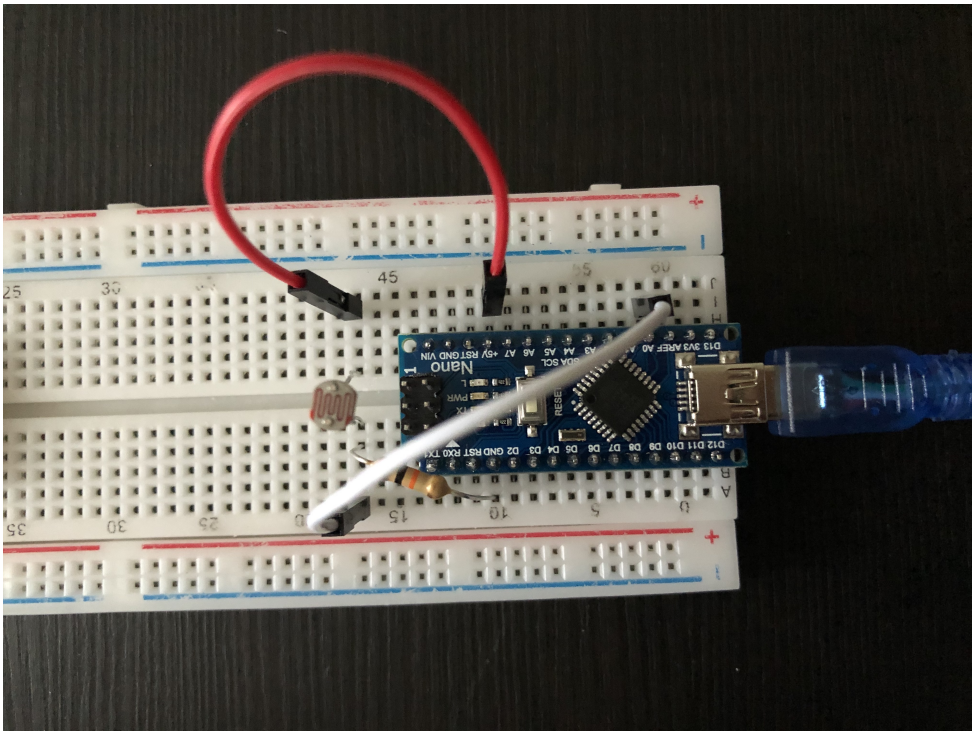

Figure 2: Light reading - circuit.

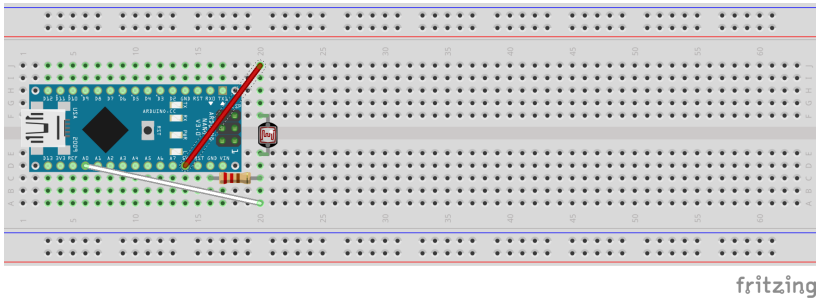

Figure 3: Light reading - schematic.

1.2 Main Task 2

Task: Use a thermistor with a low-pass filter to make temperature measurements. Print raw data by Arduino. (0-1023)

Pictures: As before, the following photos show a thermistor reading, its circuit and schematic.

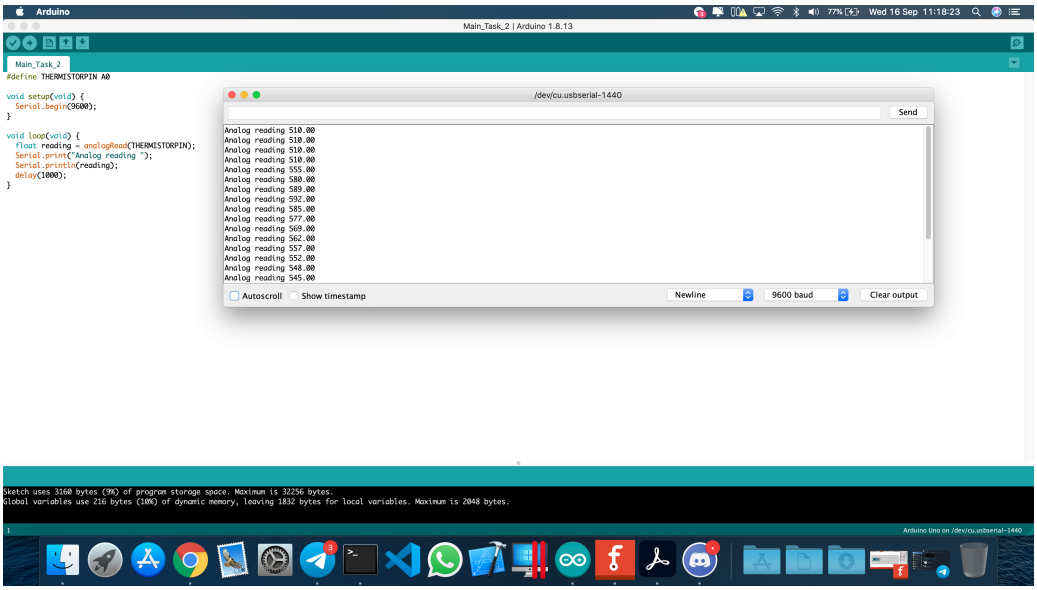

Figure 4: Temperature reading - console.

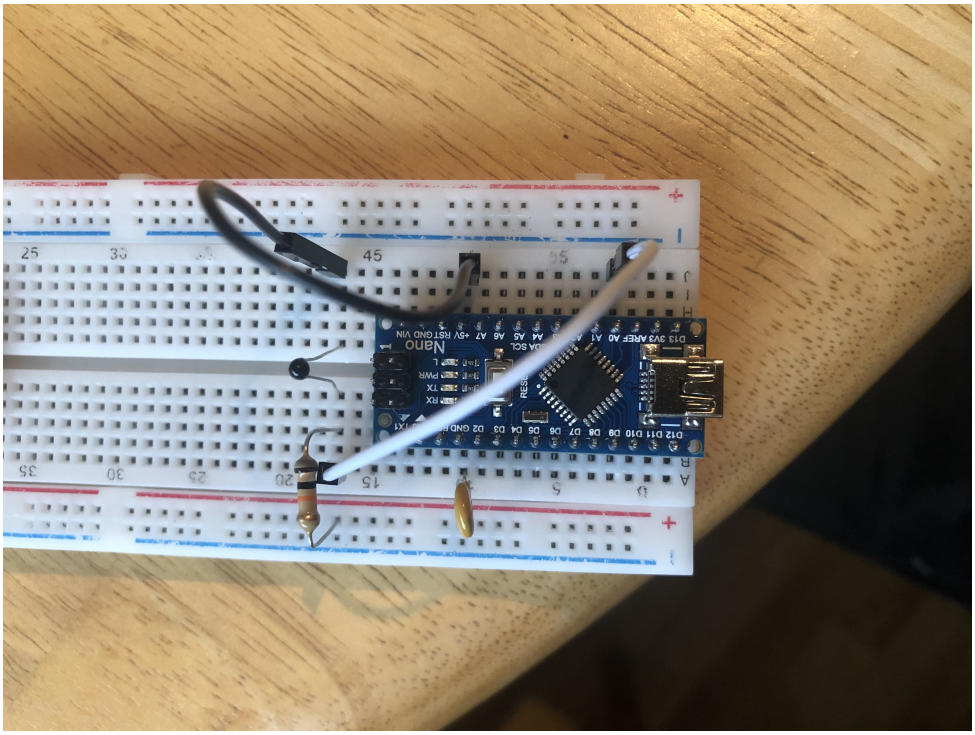

Figure 5: Temperature reading - circuit.

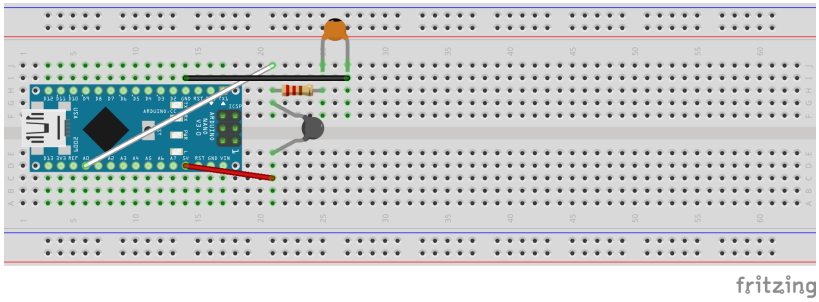

Figure 6: Temperature reading - schematic.

1.3 Main Task 3

Task: Send the data from the two sensors to the RPi and print it out on the RPi terminal.

Video: Once we merged the two circuits, we used Serial connection to send collected data from Arduino to Raspberry Pi, as shown below.

`https://shorturl.at/goEOS`

2 Hard Tasks

2.1 Hard Task 1

Task: Using a filter, smooth and adjust analogue data of a light sensor to have a stable signal with a range from 0 to 1023.

Justify your choice of filter with respect to the data inputs.

Video and explanation: The filter we used in this task to smooth the values collected by the light sensor is the arithmetic mean. We did this because the sensor values can fluctuate a lot while measuring, and by calculating the average we are sure to get approximate but more consistent in time values: specifically, we take 16 values with a 100 milliseconds delay each and then calculate the mean on those.

`https://shorturl.at/qJXY9`

2.2 Hard Task 2

Task: Calibrate the temperature sensor, it needs to be precise up to 0.5 degree within the range of 0 to 20 degrees Celsius. Explain your process.

HINT: Setting the sensor in a bag in iced water is close enough to 0°C and putting it on known temperatures for example body heat can give you additional values.

Video and explanation: In order to achieve this result, we used the formula below, which is called "Steinhart-Hart Formula".

$$T = \frac{1}{A + B \cdot \ln \frac{R}{R_0} + C \cdot \ln^2 \frac{R}{R_0} + D \cdot \ln^3 \frac{R}{R_0}}$$

Figure 7: Steinhart-Hart Formula.

This formula allows us to calculate the precise temperature in Kelvin degrees (and then execute the conversion by just subtracting 273.15) by knowing the resistors in the circuit: the values we inserted in the Arduino sketch are thus matching the resistors we used and should be changed if changing the circuit.

Then, we just approximated the values obtained in the specified range (0-20 degrees) to a 0.5 degrees precision, and left the other values outside range as they were (no precision approximation).

`https://shorturl.at/cdtxR`

2.3 Hard Task 3

Task: Make Hard#2 into a function where you can change the calibration values.

Hint: Structs

Video: For this task we didn't actually use structs as suggested, but we just used Rust to send Arduino data about the calibration values setup via Serial, and then used Serial again to continuously get values back and print them on Raspberry Pi terminal. In the video below, we show how the values appear with 0.25 as precision and 20-30 as thresholds.

`https://shorturl.at/dANOS`

3 Advanced Tasks

3.1 Advanced Task 1

Task: Log the light and temperature data into a CSV file. Make new lines of sensor values be added every 10 seconds with a time and date stamp at the end of the line. Graph 10 minutes worth of data

Screenshot and explanation: We used Rust library OpenOptions to manage write to files, and then left the program running for more than 10 minutes. Once we came back, the file was full of data gained from measurements and we used it to plot a graph in Excel, which is shown below (the data gained has some extra "UTC" characters on the date record, which are there by mistake, and is precise to the nanoseconds on the time field, which is not needed, so we adjusted these outputs in the next task).

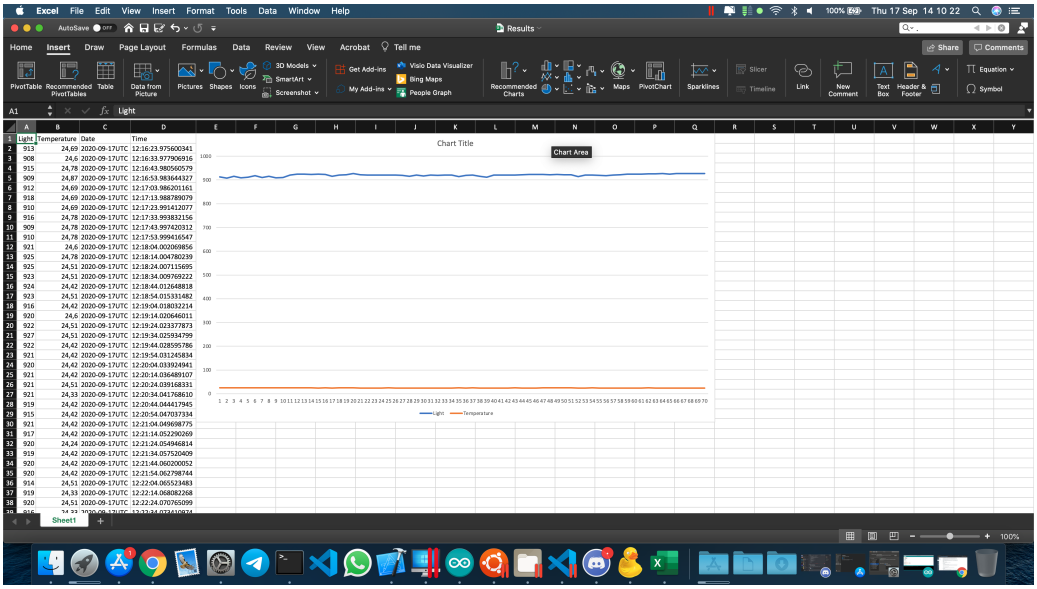

Figure 8: 10 minutes data graph.

3.2 Advanced Task 2

Task: Make the log file be automatically committed to your repository every 5 minutes.

Hint: Bash commands in Rust (but not necessarily, it is an open task)

Screenshot and explanation: To achieve this, we used Rust terminal commands support to commit and push the generated results file every 5 minutes to the lab repository, and then let the program run for some time to check if everything was working correctly. Furthermore, we fixed the "bugs"

described in the task before and made it that the commit comment contains the date and time at which the latest data was acquired.

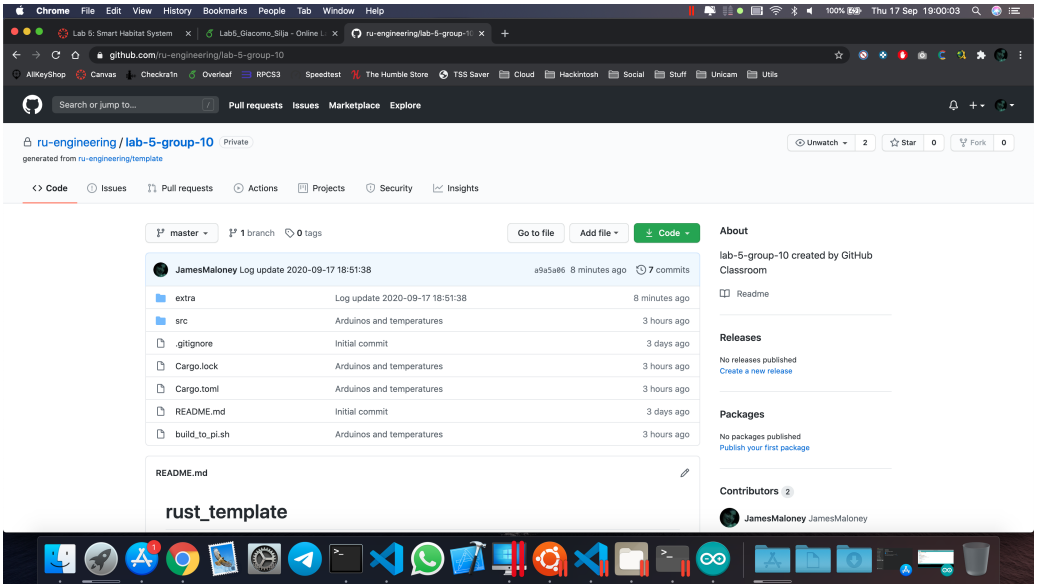

Figure 9: Automatic commit with date and time.