Exercise 1

group members:

Florian Hauser, Leon Gies, Arthur Zeidler, Sebastian Behr, Samuel Tim

## Internal RGB

Features of the Arduino Nano include an onboard RGB LED, which can be accessed via the WiFiNINA library. To control these LEDs, initialization is needed within the void setup() loop. The LEDR/G/Bs are initialized as OUTPUT.

To achieve the desired color change, three for loops are needed: one for fading from red to green, one for fading from green to blue and a final loop for fading from blue back to red. The loop iterates in steps of two, each time decrementing the previous color and incrementing the value of the following color.

The desired behavior is a fluent color change from red to blue passing green. This whole iteration should take 1 second hence each transition should take 500 milliseconds. This can be implemented by a for loop with 255 iterations and a delay of 1 millisecond after each increment.

This led to an Issue, because the LED can not change it’s value that fast. To fix this we changed the delay to two milliseconds. Now we faced the issue that the period duration is two seconds instead of one. This is solved by incrementing and decrementing the output values by two. Now the requirements are met although the transition is not as smooth as desired.

This is a necessary adjustment we had to take but does not affect the end result since you can not distinguish the different implementations in the end result.

## Thermometer

The Arduino Nano also contains an inbuilt onboard thermometer, which can be accessed via the WifiNina library. First, the Sensor is initialized as well as the serial port.

Additionally, the onboard LEDs need to be initialized and initially turned off to prevent unexpected behavior.

The void loop function includes checking the availability of reading the temperature and if so printing it to serial. Additionally we differentiated between three overlapping temperature ranges and turned on the correct LED:

The first range starts from 32° C and is not limited with an upper bound. Within this range, the red LED is turned on.

The next interval is between 36° C and 20° C. The colors of the LED is set to green.

The final interval only has an upper bound of 25°C. The corresponding color is blue.

In the overlapping intervals both LEDs are turned on.

To verify only the correct LEDs are turned on at the beginning of each iteration all LEDs are turned off. This leads to a short duration in which the LED is completely off but as figured out in the previous exercise the LED can not visualize such short timed color changes and so this is not an issue.

These checks are done in a loop with a check interval of 1 second. This is achieved by adding a sleep call to the end of the loop.

A major difficulty did not arise within the exercise itself, but rather the setup of the Arduino. The sketches would not upload after compiling, indicating an issue with the connection of the Arduino to the computer. The issue resolved itself after we asked for a spare Arduino as we physically were not able to establish a connection. A few minutes later we received the initial Arduino back and could test the temperature program as well.

## Microphone

The third exercise used the onboard microphone. The goal was to visualize the current sound input in a graph. We used the code from the described tutorial and ran it on the Arduino.

To test how accurate the microphone records we played different individual frequencies. The microphone had a lot of noise in the recording, but it could still differentiate quite well between the frequencies.

## Gyroscope

The final exercise combined a few aspects from the previous exercises. We used the output of the thermometer to check if the temperature is within a valid range. Additionally, the onboard gyroscope and acceleromometer are used to determine the rotation of the Arduino.

The correct calculation of data produced by the acceleromometer and gyroscope is a demanding task, for which we used the MadgwickAHRS library. This library converts raw data from the sensor to usable values.

Difficulties we faced included setting the status LED correctly, as it needs to turn red if at least one of many sensors output undesired data. This issue was fixed by introducing a flag variable that is initialized at the beginning of each iteration with the value “false”. It may be set to “true” if any of the measured inputs produce unwanted data e.g. too low or high temperature. If no inputs show unwanted data, the flag remains “false“ and the led stays green.