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 desired new color. We tried setting the step value to 1, but the LEDs weren´t able to change color that quickly. Also, as we settled to a step value of 2, a delay of 2 ms is needed to achieve the fading interval to green of half a second, because 255 values devided by 2 steps means we have 177 iterations. Each iteration implements a delay of 2ms, resulting in the duration of the loop adding up to 255ms. Two of these loops result in a change to green, so 255ms\*2 equals the desired half-a-second green passthrough.

## Thermometer

The Arduino Nano has 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, printing it to serial and then differentiating between three overlapping temperature ranges:

The first range starts from 32° C and is not limited with an upper bound. Within this range, the LEDs set their color to only red.

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

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

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 current sound input in a graph. We used the code from the described tutorial and ran it on the Arduino. We played a sine wave sound and were able to visualize the frequency quite well.

## 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 accelerometer are used to determine the rotation of the Arduino.

The correct calculation of data produced by the accelerometer 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.