LabVIEW II project work

Arwed Paul Meinert
42321

Lecturer: Dr. Lustermann

Winter semester 21/22

Delivery: 01.02.2022

Content

Introduction

To create an independent project in the elective LabVIEW II. The goal of the course is to use an external device as input or output e.g. for data recording or as a display element. This can be, for example, the peripherals of the computer running LabVIEW, or a single board computer (e.g. Raspberry Pi) or a microcontroller such as an Arduino.

The LabVIEW program should be able to store measured values on the computer with associated units or recording time. In this context, attention should be paid to the Producer- Consumer design pattern.

Objective of the project

The topic of our project was up to us. My idea was to display climate values, such as temperature and humidity, which are recorded by a sensor, in a LabVIEW program. The values should be displayed on a 10x10 matrix of Adafruit neopixels. The brightness should be adjustable via the VI.

Temperature and humidity values are stored by the program with a time stamp locally on the computer and displayed on the LabVIEW VI front panel in a graph. This graph is also to be shown on the matrix. The two data of humidity and temperature are to be displayed on the y-axis in different colors. The x-axis corresponds to the time. In order to adapt the recording of the data to the respective application, it should be possible to set the minimum time between two entries in order to keep the number of entries manageable over a long period of time.

Realization of the project

The Neopixel LED matrix and the humidity and temperature sensor are controlled with the help of an Arduino Nano. The Arduino is connected to the computer via the USB port and can thus send and receive the data via the serial interface.

I decided to implement the program as a state machine. For this the following states are necessary:

-Init

-Data writing

-Letter Temp +Hum

-Letter Time +Date

-Diagram Plotting

In the "Init" state, check whether a connection to the periphery is defined. Meanwhile, make sure that the "Measured values" folder is present in the VI directory. If this is not the case, it must be created.

In the "Data Write" state, the measured values of the DHT-11 sensor are stored in a file on the computer in a predefined manner with a time stamp.

The state "Ticker Temp and Hum" displays values of temperature and humidity on the LED array. If the text to be displayed exceeds 10 pixels (the width of the matrix), the values should disappear to the left as in a ticker and thus display the entire number and its respective unit.

In the "Temp and Hum ticker" state, the time (hours and minutes) is displayed in a similar way.
and the date (month and day) are displayed.

In the state "Graph Plotting" the recorded data points are to be displayed in a meaningful form on the LED matrix. The graph is updated on the VI with the recorded new data points.

The program always starts in the "Init" state. After a time it jumps to the next state. Single
States can also be skipped.

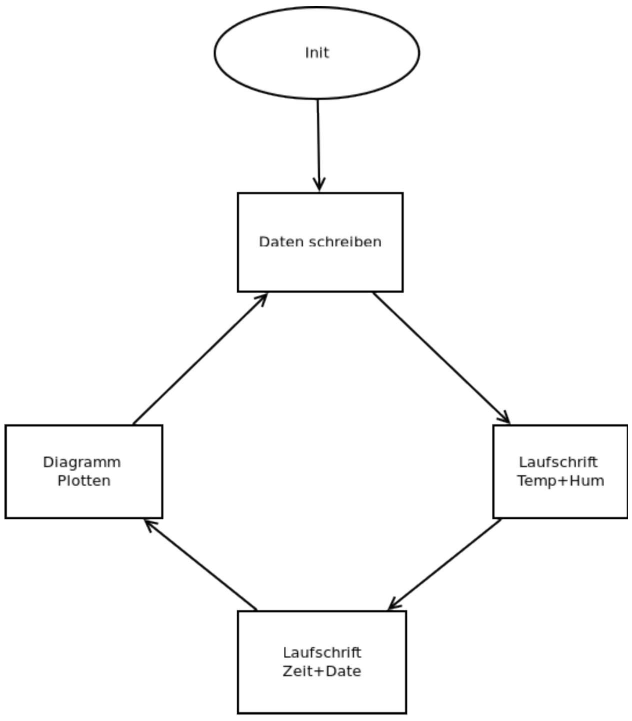

Figure 1; State machine sequence

There is possibility to exit the program at the end of each state.

Hardware project planning

The project uses an Arduino Nano V3, which is powered by the PC via a USB connection. The Arduino is plugged onto a bread board to facilitate the construction of the circuit. The potentials GND and +5V are connected to the respective busbars of the board. Here the operating voltage of the DHT-11 sensor and the WS2812 neopixel is tapped (Figure 2).

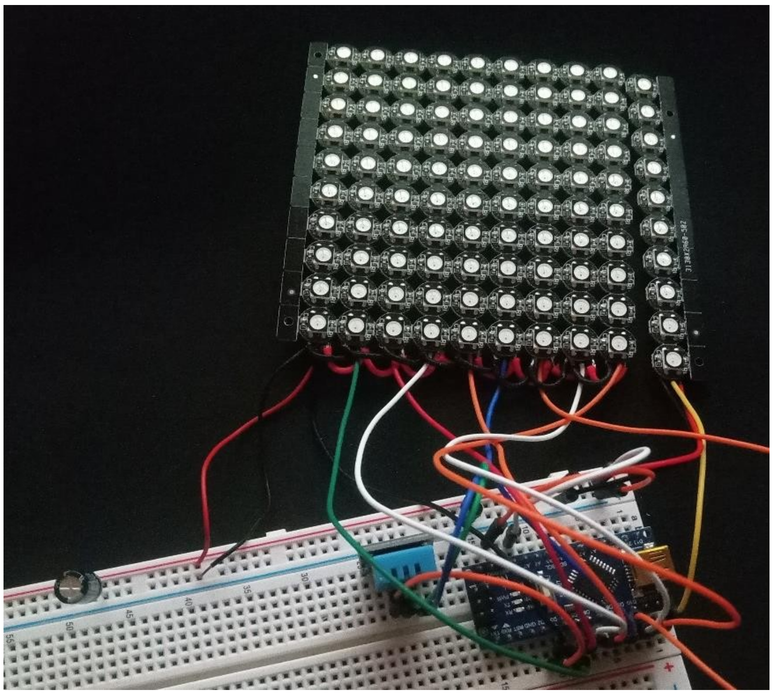

Figure 2; Structure of the circuit

To minimize potential voltage drops due to excessive current peaks of the neopixels, a 100µF capacitor is connected between +5V and GND. Depending on the desired display on the matrix, it may be useful to use an external power supply for the voltage supply.

Any number of neopixels can be connected in series via the "$_{\text{Din}}$" or "$_{\text{Dout}}$" connector, so that only one digital connection is required for any number of neopixels. This simplifies the wiring effort compared to a classic LED matrix. In this project, I always have 10 neopixels in series and connected to one digital pin of the Arduino. I decided to use D3 to D12 here.

The DEBO DHT-11 sensor is a digital sensor. I have connected it to D2 in this case.

Program of the Arduino

For the control of the Neopixel the library "Adafruit Neopixel" (v.1.1.4) must be included. The sensor needs the "DHT sensor library" (v.1.4.2). At the beginning all necessary pins are defined. In the program section "Setup" the connection to the sensor is started. Afterwards an array is created over which every single neopixel can be controlled individually. Furthermore, a serial connection with the baud rate of 9600 Bd must be initialized.

In the program section "loop" the serial interface is read. If the serial buffer contains the character 'd' it should get Arduino RGB values for the pixels. In addition, the program expects the character 's' at the beginning of each new line and the character '\r' is expected before each RGB value. Now, with the Serial.parseInt() function, the first Int value is set as the Red portion. The same happens with the values of the green and the blue part of the pixel color. After all values have been successfully received and displayed, the Arduino program sends the character 'e' over the serial bus.

After receiving the character 'q', the temperature and the humidity of the sensor are read out via the respective commands. These values are written to the serial bus. The recorded values are separated and distinguished by '\r'. After the successful sending of the climate data ‚e' is written to the bus as with the RGB value recording.

LabVIEW program

In order to make the main program clearer, a large part of the programs has been outsourced to so-called sub-VIs. Each sub-VI is a single function with its own inputs and outputs, in which other sub-VIs can be called.

Serial Write:

In the VI the RGB values of a 10x10 array are written to the Arduino via the serial interface. For this purpose a VISA resource name must be specified. Furthermore the intensity can be defined. Additionally the array to be displayed must be passed. It consists of an array of 10x10 U32 values. The respective values of the individual pixels are read out via two for loops and written to the serial interface. The pattern that the Arduino expects to receive is used. So a 'd' at the beginning, before each new row a 's' and between each value a '\r'. A while loop is also used to check if there are bytes in the serial buffer at the VISA port. In this case, the Arduino program has received all RGB values and sent the 'e' to the LabVIEW program. Otherwise, the array is sent again until it is completely received. At the end the serial connection is closed. So other program parts can access the serial port.

Serial reading:

This VI is used to read the temperature and humidity values. First the string 'q' is sent to the Arduino, whereby the Arduino writes the values into the serial interface. A property node is used to check if there are still bytes available at the defined VISA port. If this is the case, the serial interface is read out. If several strings are read, they are concatenated. If no more bytes are available in the buffer, the whole string is searched for '\r'. This character separates temperature and humidity. At the end of the string there must be the character 'e'. This is the sign for the Arduinos that all characters have been transmitted successfully. Also in this VI, the serial connection is closed after the loop is completed. The values for temperature and humidity are declared as outputs of the VI.

Write measured values :

Here the measured values, which are read out in the "Serial Read" subVI, are saved in a .txt file. If a path is specified as input, a folder "Measured values" is created in the directory and a file with the format yyyymmdd.txt is created there. If no path is specified, the VI uses the folder in which it is saved. If a file with the same name exists, it will be opened. At the end, each time the VI is called, a timestamp, the temperature and the humidity are saved to the end of the file. The three values are separated with a tab. Also, the temperature and humidity values are written to the global variables of the same name. The VI has a file ref number as output. This allows other VIs to access the file very easily. Before the VI is exited, the file is closed. This avoids that the file is accessed by different program parts at the same time.

Color Scaler:

The goal is to output a color depending on the input, which results from two different colors. There is the possibility to pass an upper color and a lower color to the sub-VI. In addition, two numbers, a lower limit and an upper limit can be passed. Depending on where the input value lies relative to the two number limits, a color is set as output that is a mixture of both colors. So if the input is exactly the lower boundary the returned

Color is the lower color limit. If the input is equal to the upper number limit, the output color is equal to the upper color limit. If the input is exactly between the upper and lower limit, the output color is 0.5*lower color limit+0.5*upper color limit.

If no values are entered in the input windows of the color and number limits, the default values are used. You can switch between temperature (false) or humidity (true) with the Boolean flag.

Create Representing Measured Values :

In the VI, 10 values are extracted from the measured values, which can then be displayed on the array in a later VI. For this purpose, the path or the reference number under which the measured values are stored must be specified. In the VI, the columns of temperature and humidity are written to individual arrays. The number of measured values is calculated. Because the array that is output may only have 10 rows, the points at which the temperature and the humidity must be determined must now be calculated. It is assumed that the measured values were taken at even intervals.

To achieve this, the number of rows in the measured value file is divided by 10. The result is then multiplied in a for loop with the loop iteration. In the case where there are exactly 10 values in the file, the result is 1 measurement/pixel. However, it is also possible that less than 10 measured values were recorded in the table. In this case, the number of measured values/pixel is <1. In the case that there are more than 10 measured values, the number of measured values/pixel is>1. Thus, in the case that there are not exactly 10 measured values in the file, intermediate steps in the measured values must also be calculated. For example, if there are only two measured values, the first measured value should be displayed on the first pixel, the second measured value on the last pixel. In between, however, values should also be displayed.

To achieve this, I decided to perform a spline interpolation with the measured values. In a spline interpolation, two measured values n and n+1 are interpolated with a third degree polynomial[1] . This provides very good results especially with less than 10 measured values. However, in contrast to an approximation, it also has the advantage that one does not have to know beforehand which degree the compensation line should have. The Y-values are respectively the arrays of the temperature and humidity. The x-values are where the y-values are (it is assumed that all values are equally spaced). As xi the array of the measured values/pixels is passed. This results in 10 values each for temperature and humidity, which are also displayed in an array at the end.

Plot measured values :

Here the interpolated values of temperature and humidity in a 10x10 array are represented by different colors. The smallest value is displayed in the lowest row with the respective lower color limit. The highest value is displayed in the top row and gets the value of the upper color limit. All values in between are displayed in the area in between with the colors that the ColorScaler Sub-VI calculates. This happens for the temperature and for the humidity. At the end both arrays are merged and displayed.

Create Wavechart :

In the VI, the measured values for temperature and humidity are converted into a form that they can be displayed by an XY graph on the control panel of the main VI. As input of the sub-VI the path or the reference number of the measured values is needed. The file is searched for strings, which are written into an array. The first column is the timestamp, the second the temperature and the third the humidity. The XY graph needs as input one or more clusters with x and y values. In this case, timestamps are used as y-values. To get from

[1] Cf. Bronstein, Pocketbook of Mathematics, p.956

to generate a timestamp from the string of the first column, each row is searched for a string in the time format '%<%c>T '. Each value is auto-indexed and combined with the respective one-dimensional arrays of the temperature column and the humidity column to form two clusters. An array is again created from these two arrays. This array can then be displayed.

Decimal- Array:

In the VI, an input of a number and a string is written into a three-dimensional array in which each array level represents a character in a 5x3 array of colors.

When a number is passed to the VI, it is converted to a string. The passed string is appended to the string of the number. If no number is passed, the passed string stands alone. Each character in the string is compared to an enum constant where all existing characters are stored. If there is a match with a character, the respective state is called in which a 5x3 matrix is stored. In the matrix each field, which is to be represented in the color array in the respective color, is marked with a one. Now every cell of the matrix is scanned. If there is a one, the color is written into the cell of the array. Each new character is written into a new layer of the array. If each character of the string was processed, is written to the array is returned. Because for each character only once a matrix and a new state in the enum constant "ZahlenEnum" must be created, it is very easy to add new characters. So far I have added the characters [0,1,2,3,4,5,6,7,8,9,:,%,C,°,.] necessary for my application.

Num_Str_Together:

In the VI, a number and a string become an array of size 5xn, in which the number and the string are displayed in pixels of the color of the Color Box. To do this, a three-dimensional array is created with the Decimal Array sub-VI. Then each layer is written to the end of a separate matrix. One column is left free after each character. The resulting array is output.

Disp all:

In this VI, either the temperature and humidity data or the time and date are displayed on the LED matrix. In both cases more than 10 pixels are required for the display, therefore it is necessary that the font scrolls. It is also possible to set the time that is waited between an update of the matrix.

If there are values for temperature and humidity, the ticker is displayed for both values. If both values are zero, it is assumed that the time and date should be displayed. In the case of temperature and humidity, the sub-VI "Num_Str_Together" is used to append the unit '°' to the value of temperature and the character '%' to the humidity. The resulting arrays are displayed on top of each other in two different colors. In the case that only the time and date are to be displayed, the program expects only strings. The color of the time is set with red, the date with green. The program checks how long the array is and initially displays the rows 0-9 on the LED matrix. After the waiting time is over and rows 2-11 are displayed and so on. For this the sub-VI "Serial write" is used

It is not necessary to set a waiting time, because it takes a relatively long time until the Arduino has received and displayed all values.

ScrollingTempHum:

In the VI the values of temperature and humidity are displayed on the LED matrix. For this purpose, the sensor values are read out via the "Serial Read" sub-VI and the necessary colors are assigned to the resulting values via the "Color Scaler" VI. The standard limits for the colors and the numbers are used for this. To the "Disp all" sub-VI the two calculated colors and the values of the temperature and the humidity are transferred. In addition, the desired intensity is transferred.

Scrolling text clock:

Here the current time and date are displayed on the LED matrix. For this purpose, the time with the format string '%I:%M' and the date with '%d.%m' are passed to the sub-VI as well as the desired brightness.

Main program:

As usual with a state machine, the "Init state" is passed as the initial state. From here
the individual states run one after the other.

In the "Write data" state the sub-VI "Write data" is called. This VI writes both the path and the Ref number of the measured values into a shift register. Depending on the switch position on the front panel the next step is "Ticker Temp+Hum" or "Ticker Time u Date".

In the states "Ticker Temp + Hum" and "Ticker Time u Date" the respective sub-VIs are called.

In the "Plot diagram" state, the measured values from the file are displayed on the LED matrix via the "Plot measured values" sub-VI. Via a sequence structure, the sub-VI is then plotted.
"Create Wavechart" is executed and the resulting array is displayed on an XY graph. In the state also the adjustable minimum time between measured values is set with the Express-VI
"Elapsed Time" is compared. If the time has not yet elapsed, the state "Write data" is set.
skipped.

Application of the project

The aim of the project was to use the program to measure the humidity and temperature over a closed time frame. In doing so, one should always be able to read the relevant information and the previous trend of the recorded courses on the LED matrix.

Application scenario

To test the function of the program, I decided to measure the climate values during an indoor workout on the bike. The bike is mounted on a roller trainer, which automatically adjusts the resistance on the rear tire. In the case under consideration, the workout lasts 90 minutes and an average power of 315W is to be produced (see Figure 3).

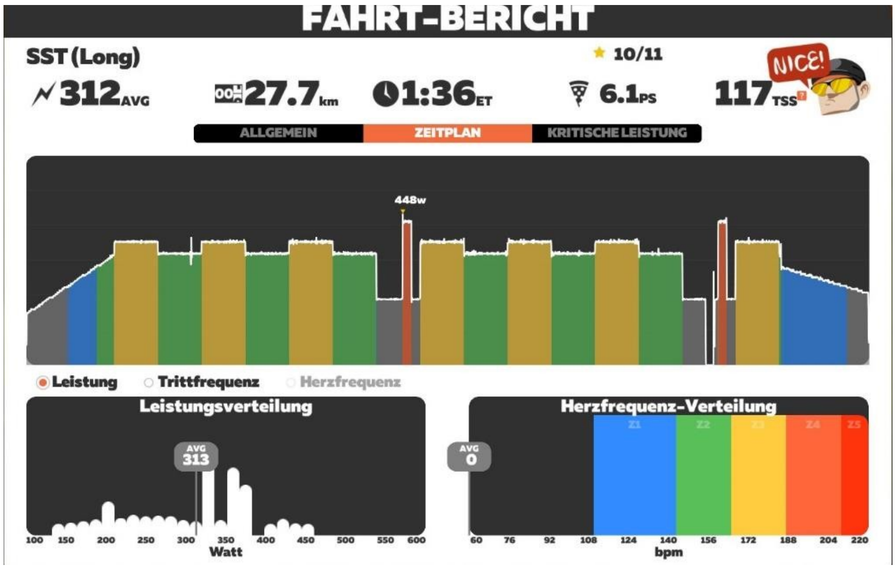

Figure 3; Power progression of the workout

During the workout the window is open. Also, two fans are turned on for better cooling.

It can be expected that the air humidity will increase after a short time due to transpiration. It can also be expected that the room temperature will also rise due to the heat loss from the roller trainer and due to an increase in body temperature.

Implementation

The front panel of the VI was open during the entire execution. Due to the foreseeable length of the data recording, the minimum time between data points was set to 30 seconds. The display of temperature and humidity on the front panel provided a good indication of the current values. However, the LED matrix was also useful. On it, the temperature (top) and humidity (bottom) were displayed at periodic intervals (Figure 4). In the example, the temperature is 20°C and the humidity is 63%.

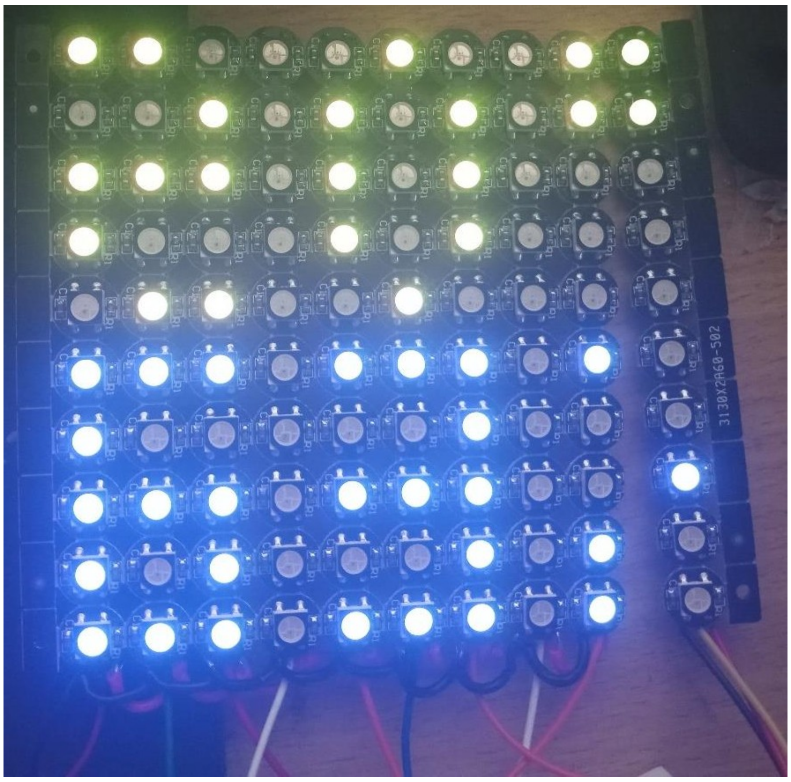

Figure 4; plot of temperature (yellow) and humidity(blue).

One also had a good overview of the respective tendencies due to the representation of all previous measuring points or the interpolated intermediate steps:

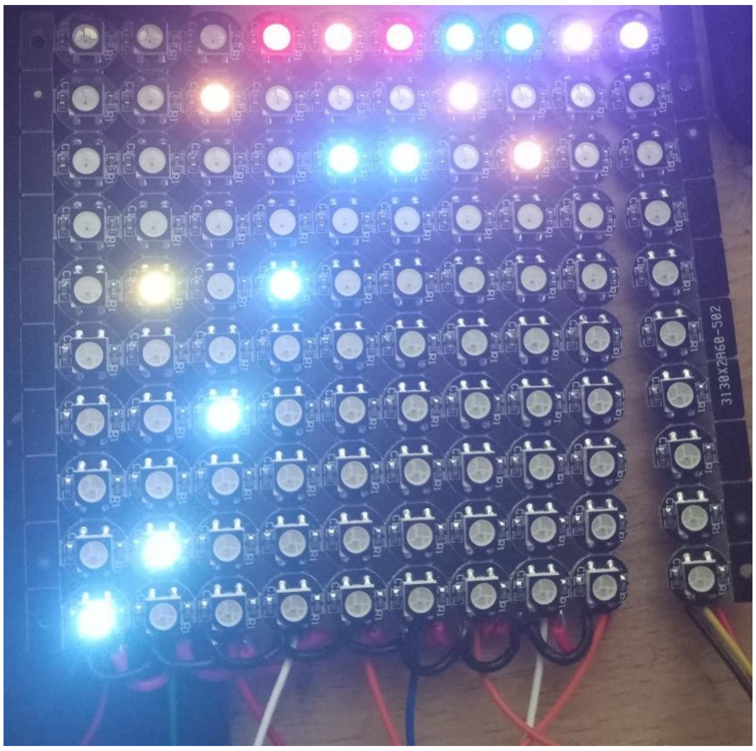

Figure 5;Representation of the time course of the data

It can be seen that the humidity (blue) has its minimum at the beginning of the measurement. It increases continuously over the course of the measurement of 90 minutes and has its maximum value at the end (last column).

The temperature fluctuates over the course of the measurement. Especially at the temperature you can see that not only the height at which the pixels shine allows a statement about the temperature, but also the strength of the color. In the case of temperature, the maximum temperature is a pure red. If the color is more orange the temperature is lower. So even if there are several values in the top row, you can still make a statement about at which time a higher temperature was measured. This measurement also illustrates what happens when the temperature and humidity are displayed on the same pixel. In the first and in the last two columns only one LED is on. Especially in the last two columns, however, you can see that it is neither a pure blue nor a pure red. Rather, it is a mixture of red and blue; that is, violet.

With the help of this representation one can only estimate a trend of the courses. It is not possible to make a statement about actual temperatures. Therefore, if you want to analyze a higher resolution and absolute measured values over time, it is better to look at the plot on the front panel:

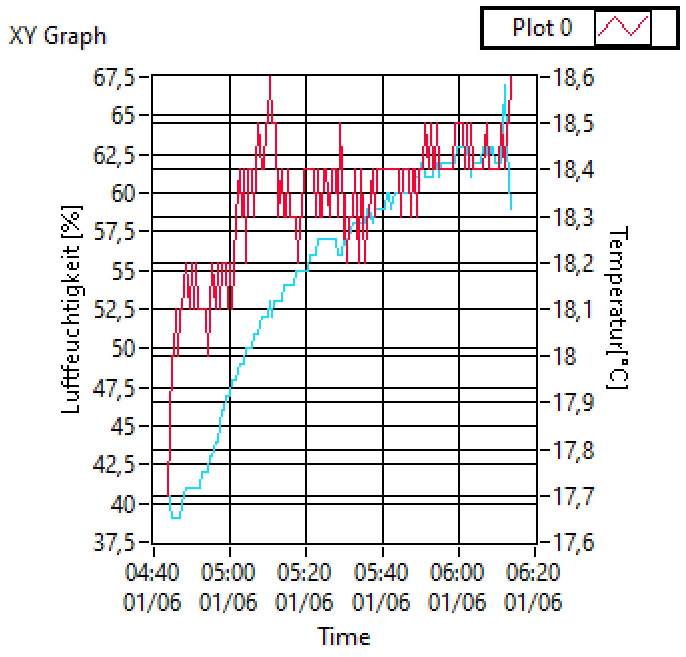

Figure 6; XY Chart

This is the display on the front panel. Here the respective data points of temperature (red) and humidity (blue) are plotted by time. There is also an axis label. You can immediately see that the humidity actually increases from a value of about 42% at 4:40 to a final value of about 66% at 6:20. This course has been reflected very well by the display on the LED matrix.

The temperature, on the other hand, also increases over time, but the measured values are only between 17.7°C and 18.6°C. The trend shown on the LED matrix is therefore not very representative or misleading. The display with the minimum value in the lowest row and the highest value in the top row gives the impression that the temperature values span a wide range of values. In addition, the swings in temperature are very large relative to the overall behavior. Thus, a maximum can easily result from an erroneous reading. At the end of the 90 minutes, a total of 155 data points were entered into the file
"Measured values220106.txt" written.

Conclusion on the function

In the application of the program it is important to consider the display on the LED matrix only as a tendential value. If the absolute temperature or humidity leads one to a certain point in time, the display on the front panel is to be used. In order to see the current temperature, humidity or the previous course at a glance, the LED matrix offers a very good possibility. In the end, one should always ask oneself before a measurement, which measured values are to be expected and which display form is suitable for the question. In this application example, it was more important to recognize the trends as quickly and easily as possible, and in this respect the display format on the LED matrix proved to be very convenient.

Potential errors

Producer-Consumer Design Patterns:

The producer-consumer design pattern ensures that multiple processes do not access data simultaneously and perform different operations (read/write). This could lead to conflicting data[2] .

In the project there are two possible places where problems could occur. Two or more VIs could access the serial interface at the same time and thus hinder the data exchange or make the sent data unusable. Also two program parts could access the measured value file.

Because the program is executed as a state machine, i.e. there are no two loops running in parallel, care must be taken that no attempt is made to open a serial connection twice in any state and to close the connection again after the data exchange has been completed. In the program it comes at no time to the fact that at the same time two serial connections could be established.

It must also be ensured that only one program executes an operation with the measured values file at a time. In the "Write measured values" state, only one operation is performed on the file. In the "Plot measured values" state, on the other hand, simultaneous access to the measured values file could occur. Therefore the sequence structure is used. So only the first operation (read measured values and write them to the matrix) is executed. Only when the process is finished, the file is accessed again to display the measured values in the graph on the front panel.

Power supply of the Neopixel:

In the data sheet of the WS2812 Neopixel, a current of 20mA/LED is specified per color[3] . So if the color white is displayed on the LED, the current is 60mA. In my measurement, this results in a basic current of 88mA without an LED representing a color. If now one of the LEDs is set to the color white at maximum brightness the total current is 110mA. If no external power supply is used to power the Neopixels, the current must come from the Arduino and its USB connection to the PC. The Arduino has a maximum output current of 800mA[4] . However, if the Arduino is only connected to a USB 2.0 port, the maximum current is 500mA (see USB 2.0 specification, p.178)[5] . So if no additional power supply is used, problems can occur if too many LEDs shine too bright.

[2] https://www.ni.com/de-de/support/documentation/supplemental/21/producer-consumer-architecture-in-labview0.html

[3] https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

[4] https://diyi0t.com/arduino-nano-tutorial/

[5] https://usb.org/document-library/usb-20-specification

In order to experimentally determine the maximum amount of current required, the power supply was implemented with an Elegoo Power MB V2. All 100 LEDs were set to the RGB value (255,255,255). This value was then multiplied by the brightness (0-1).

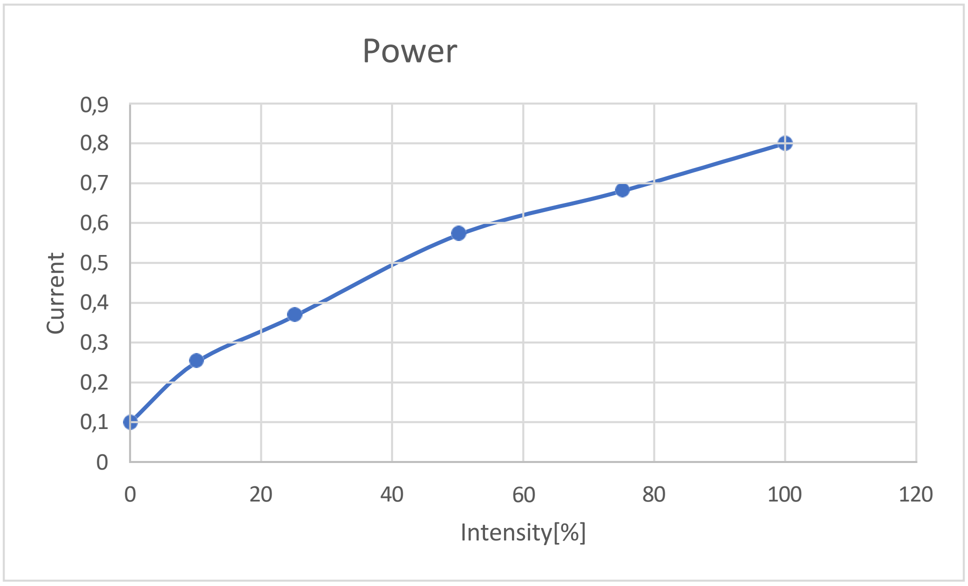

Figure 7; Neopixel power consumption

You can see that a current of 800mA is needed at maximum brightness. This current could not be provided by a USB 2.0 port and it can lead to incorrectly displayed colors. In the project, it is currently not the case that so many LEDs are switched on at the same time. Also, the brightness is set to 10% by default. However, if you implement new dressable pages, you can't ignore the power consumption.

Display of the curves on the neopixel matrix

If two measuring points are to be displayed on one pixel, the RGB values of the two colors to be displayed are added. Normally, the brightness in the front panel is set to a value that is significantly smaller than 50%. Thus, the maximum RGB value displayed can only be (127,127,127). If now in the worst case two such values are added, nevertheless still a representable color results, which can be indicated on the matrix. If the brightness is set to a value greater than 50%, parts of the RGB value greater than 255 may occur. If this is the case, the respective color will be displayed changed. For example, an RGB value of (256,256,256) will be displayed as (1,1,1). This can lead to unwanted colors, especially when "mixing".

Potential extensions

Due to the implementation of the ticker it is possible to display various measured values or texts on the LED matrix. The same applies to the front panel, of course. A large number of sensors can also be connected via the Arduino. For example, one could not evaluate the temperature and humidity data, but the humidity of floors. The LED matrix could thus indicate when the temperature falls below a certain limit.