LabVIEW II-Projektarbeit

Inhalt

Einleitung

Im Wahlpflichtfach LabVIEW II ein eigenständiges Projekt zu erstellt. Ziel des Kurses ist es, ein externes Gerät als Input oder Output z.B. zur Datenaufzeichnung oder als Anzeigeelement zu nutzen. Dies kann beispielsweise die Peripherie des Computers sein, auf dem LabVIEW läuft, oder auch ein Einplatinenrechner (z.B. Raspberry Pi) oder ein Microcontroller wie ein Arduino.

Das LabVIEW Programm sollte in der Lage sein, Messwerte auf dem Computer mit dazugehörigen Einheiten oder der Aufnahmezeit, zu speichern. In diesem Zusammenhang ist auf das Producer-Consumer-Entwurfsmuster zu achten.

Zielsetzung des Projektes

Das Thema unseres Projektes war uns selbst freigestellt. Meine Idee war es Klimawerte, wie Temperatur und Luftfeuchtigkeit, die von einem Sensor aufgenommen werden, in einem LabVIEW Programm anzeigen zu lassen. Die Werte sollen auf einer 10x10 Matrix aus Adafruit Neopixeln dargestellt werden. Die Helligkeit soll über das VI einstellbar sein.

Temperatur und Luftfeuchtigkeitswerte werden von dem Programm mit einem Zeitstempel lokal auf dem Computer gespeichert und auf dem LabVIEW VI Frontpanel in einem Graphen dargestellt. Dieser Graph soll auch auf der Matrix gezeigt werden. Die beiden Daten der Luftfeuchtigkeit und der Temperatur sollen auf der y-Achse in unterschiedlichen Farben dargestellt werden. Die x-Achse entspricht dabei der Zeit. Um die Aufzeichnung der Daten an die jeweilige Anwendung anzupassen, soll es die Möglichkeit geben, die minimale Zeit zwischen zwei Einträgen fest zu setzten, um die Anzahl der Einträge auch über einen langen Zeitraum überschaubar zu halten.

Realisierung des Projektes

Die Neopixel LED-Matrix und der Luftfeuchtigkeit und Temperatur Sensor wird mit Hilfe eines Arduino Nano angesteuert. Der Arduino ist über den USB-Anschluss mit dem Computer verbunden und kann so die Daten über die serielle Schnittstelle senden und empfangen.

Ich habe mich dazu entschieden, das Programm als State-Maschine zu implementieren. Dafür sind die folgenden States notwendig:

-Init

-Daten Schreiben

-Laufschrift Temp +Hum

-Laufschrift Zeit +Date

-Diagramm Plotten

Im „Init“ State ist zu prüfen, ob eine Verbindung zur Peripherie definiert ist. Derweil ist sicherzustellen, dass im Verzeichnis der VI der Ordner „Messwerte“ vorhanden ist. Wenn das nicht der Fall ist, ist er zu erstellen.

Im „Daten Schreiben“ State werden die Messwerte des DHT-11 Sensors in einer vorher festgelegten Art mit einem Zeitstempel in einer Datei auf dem Computer gespeichert.

Der State „Laufschrift Temp und Hum“ zeigt Werte der Temperatur und der Luftfeuchtigkeit auf dem LED-Array anzeigen lassen. Überschreitet der darzustellende Text mehr als 10 Pixel (die breite der Matrix) sollen die Werte wie bei einer Laufschrift nach Links verschwinden und so die gesamte Zahl und ihre jeweilige Einheit anzeigen.

Im State „Laufschrift Temp und Hum“ wird auf ähnliche Weise die Uhrzeit (Stunden und Minuten) sowie das Datum (Monat und Tag) dargestellt.

Im State „Diagramm Plotten“ sollen die aufgezeichneten Datenpunkte in sinnvoller Form auf der LED-Matrix dargestellt werden. Der Graph wird auf dem VI mit den aufgezeichneten neuen Datenpunkten aktualisiert.

Das Programm startet immer im „Init“ State. Nach einer Zeit Spring es zu nächsten State. Einzelne States können auch übersprungen werden.

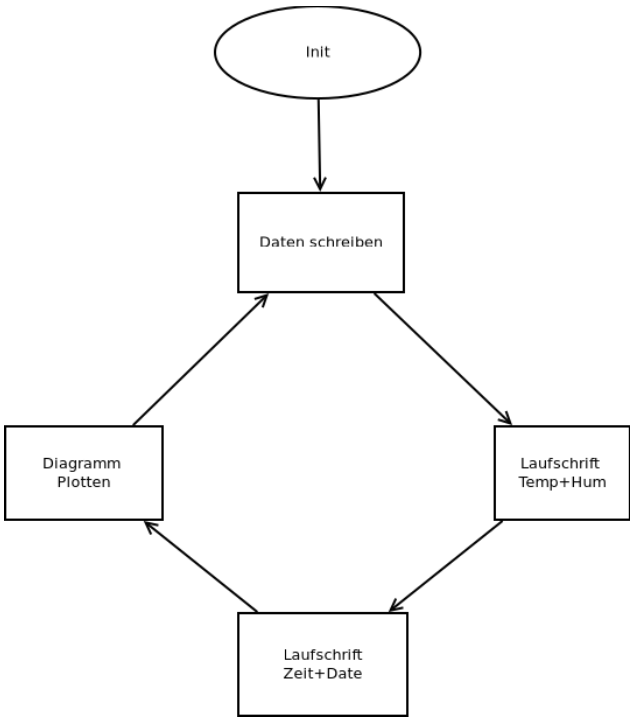

Abbildung 1; Ablauf der State-Maschine

Es gibt Möglichkeit geben das Programm am Ende jedes States zu beenden.

Hardwareprojektierung

Für das Projekt wird ein Arduino Nano V3 genutzt, der über eine USB-Verbindung vom PC mit Strom versorgt wird. Der Arduino wird auf ein Bread-Board gesteckt, um den Aufbau der Schaltung zu erleichtern. Die Potentiale GND und +5V werden auf die jeweiligen Sammelschienen des Boards geführt. Hier wird die Betriebsspannung des DHT-11 Sensors und der WS2812 Neopixel abgegriffen (Abbildung 2).

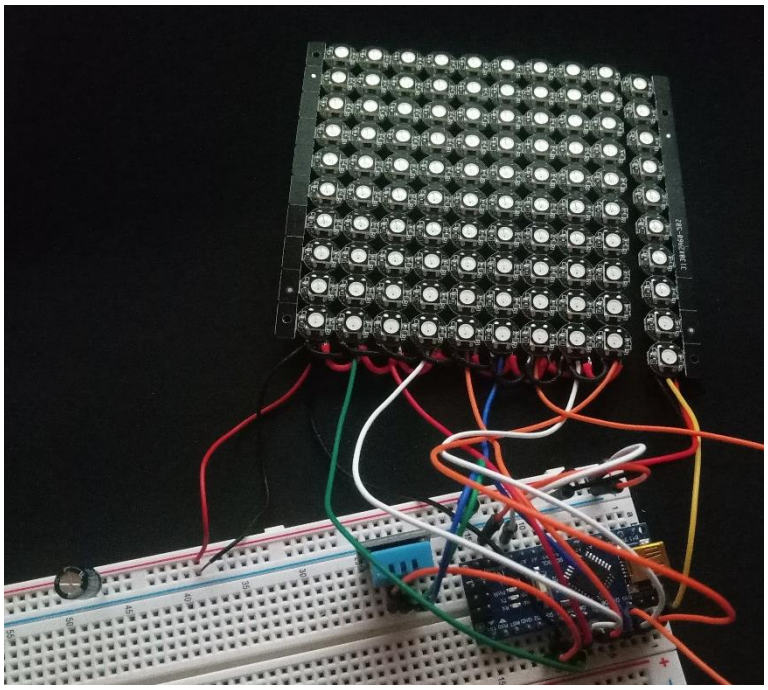

Abbildung 2; Aufbau der Schaltung

Um potenzielle Spannungsabfälle durch eine zu große Stromspitzen der Neopixel zu minimieren, wird zwischen +5V und GND ein 100µF Kondensator geschaltet. Abhängig von der gewünschten Anzeige auf der Matrix kann es sinnvoll sein, ein externes Netzteil zur Spannungsversorgung zu nutzen.

Es können beliebig viele Neopixel über den „D_{in}“ bzw. „D_{out}“ Anschluss in Reihe geschaltet werden, so dass für beliebig viele Neopixel nur ein digitaler Anschluss notwendig ist. Das vereinfacht den Verdrahtungsaufwand im Vergleich zu einer klassischen LED-Matrix. In diesem Projekt habe ich immer 10 Neopixel in Reihe und an einen Digitalpin des Arduino angeschlossen. Ich habe mich hier für D3 bis D12 entschieden.

Der DEBO DHT-11 Sensor ist ein digitaler Sensor. Ich habe ihn in diesem Fall an D2 angeschlossen.

Programm des Arduinos

Für die Steuerung der Neopixel muss die Bibliothek „Adafruit Neopixel“ (v.1.1.4) eingebunden werden. Der Sensor benötigt die „DHT sensor library“ (v.1.4.2). Am Anfang werden alle notwendigen Pins definiert. Im Programmabschnitt „Setup“ wird die Verbindung zu dem Sensor gestartet. Anschließend wird ein Array erstellt über das jedes einzelne Neopixel individuell angesteuert werden kann. Außerdem muss eine serielle Verbindung mit der Baudrate von 9600 Bd initialisiert werden.

Im Programmabschnitt „loop“ wird die serielle Schnittstelle gelesen. Ist im Seriellen Buffer das Zeichen ‚d‘ soll er Arduino RGB-Werte für die Pixel erhalten. Außerdem erwartet das Programm zu Beginn jeder neuen Zeile das Zeichen ‚s‘ und vor jedem RGB-Wert wird das Zeichen ‚\r‘ erwartet. Mit dem Befehl $\mathrm{Serial.parseInt()}$ wird nun der erste Int-Wert als Rot Anteil gesetzt. Das Gleiche geschieht mit den Werten des Grün- und des Blauanteils der Pixelfarbe. Nachdem alle Werte erfolgreich empfangen und angezeigt wurden sendet das Arduino Programm das Zeichen ‚e‘ über den Seriellen-Bus.

Nach dem Empfangen des Zeichens ‚q‘ wird die Temperatur und die Luftfeuchtigkeit des Sensors über die jeweiligen Befehle ausgelesen. Diese Werte werden auf den Seriellen Bus geschrieben. Die aufgezeichneten Werte werden durch ‚\r‘ voneinander getrennt und unterschieden. Nach dem erfolgreichen Senden der Klimadaten wird wie bei der RGB-Wert-Aufzeichnung ‚e‘ in den Bus geschrieben.

LabVIEW Programm

Um das Hauptprogramm übersichtlicher zu gestalten, wurde ein Großteil der Programme in sogenannte Sub-VIs ausgelagert. Jedes Sub-VI ist eine einzelne Funktion mit eigenen Ein- und Ausgängen, in denen andere Sub-VIs aufgerufen werden können.

Serielles Schreiben:

In dem VI werden die RGB-Werte einer 10x10 Array über die Serielle Schnittstelle an den Arduino geschrieben. Dazu muss ein VISA resource name angegeben werden. Außerdem kann die Intensität definiert werden. Zusätzlich muss der darzustellende Array übergeben werden. Sie besteht aus einem Array aus 10x10 U32 Werten. Die jeweiligen Werte der einzelnen Pixel werden über zwei for-Schleifen ausgelesen und an die serielle Schnittstelle geschrieben. Dabei wird das Muster verwendet, das der Arduino zum Empfangen erwartet. Also ein ‚d' am Anfang, vor jeder neuen Reihe ein ‚s' und zwischen jedem Wert ein ‚\r'. In einer While-Schleife wird außerdem geprüft, ob an dem VISA Port Bytes im seriellen Puffer vorhanden sind. In diesem Fall hat das Programm des Arduinos alle RGB-Werte erhalten und das ‚e' an das LabVIEW Programm gesendet. Ansonsten wird das Array noch einmal gesendet, bis es vollständig empfangen wurde. Am Ende wird die Serielle Verbindung geschlossen. So können andere Programmteile auf die serielle Schnittstelle zugreifen können.

Serielles Lesen:

Diese VI wird zum Auslesen der Temperatur und Luftfeuchtigkeit Werte genutzt. Zunächst wird der String ‚q' an den Arduino gesendet, wodurch der Arduino die Werte in die Serielle Schnittstelle schreibt. Über einen Eigenschaftsknoten wird geprüft, ob noch Bytes am definierten VISA Port verfügbar sind. Ist das der Fall wird die serielle Schnittstelle ausgelesen. Kommt es dazu, dass mehrere Strings ausgelesen werden, werden diese aneinandergehängt. Sind keine Bytes mehr im Puffer verfügbar wird der gesamte zusammengesetzte String nach ‚\r' durchsucht. Dieses Zeichen trennt Temperatur und Luftfeuchtigkeit. Am Ende des Strings muss das Zeichen ‚e' stehen. Das ist das Zeichen für den Arduinos, dass alle Zeichen erfolgreich übertragen wurden. Auch bei diesem VI wird die serielle Verbindung nach Abschließen der Schleife geschlossen. Die Werte für Temperatur und Luftfeuchtigkeit sind als Ausgänge des VIs deklariert.

Messwerte schreiben:

Hier werden die Messwerte, die in der SubVI „Serielle Lesen" ausgelesen werden, in einer .txt Datei abgespeichert. Ist als Eingang ein Pfad angegeben, wird in dem Verzeichnis ein Ordner „Messwerte" erstellt und dort eine Datei mit dem Format jjjjmmtt.txt angelegt. Ist kein Pfad angegeben nutzt die VI den Ordner, in dem sie selbst abgespeichert ist. Existiert eine Datei mit dem gleichen Namen, wird sie geöffnet. Am Ende wird bei jedem Aufrufen des VIs an das Ende der Datei ein Zeitstempel, die Temperatur und die Luftfeuchtigkeit gespeichert. Die drei Werte sind mit einem Tabulator getrennt. Außerdem werden die Werte für Temperatur und Luftfeuchtigkeit an die gleichnamigen globalen Variablen geschrieben. Die VI hat eine File Refnummer als Ausgang. So können andere VIs sehr einfach auf die Datei zugreifen. Bevor das VI verlassen wird, wird die Datei geschlossen. So wird vermieden, dass gleichzeitig auf die Datei von verschiedenen Programmteilen zugegriffen wird.

Color Scaler:

Ziel ist es je nach Input eine Farbe auszugeben, die sich aus zwei verschiedenen Farben ergibt. Es gibt die Möglichkeit eine obere Farbe und eine untere Farbe an das Sub-VI zu übergeben. Außerdem können zwei Zahlen, eine untere Grenze und eine obere Grenze übergeben werden. Je nachdem, wo der Input wert relativ zu den beiden Zahlengrenzen liegt, wird eine Farbe als Output gesetzt, die eine Mischung aus beiden Farben ist. Ist der Input also genau die untere Grenze ist die zurückgegebene

Farbe das untere Farblimit. Ist der Input gleich dem oberen Zahlenlimit ist die Output Farbe gleich dem oberen Farblimit. Ist der Input genau zwischen oberer und unterer Grenze ist die Output Farbe 0,5*unteres Farblimit+0,5*oberes Farblimit.

Wenn in die Eingabefenster der Farb- und Zahlenlimits keine Werte eingetragen werden, wird auf die Standartwerte zurückgegriffen. Man kann mit der Booleschen Flag zwischen Temperatur (false) oder der Luftfeuchtigkeit (true) umschalten.

Darstellende Messwerte erstellen:

In dem VI werden aus den gemessenen Werten 10 Werte extrahiert, die in einem späteren VI dann auf dem Array dargestellt werden können. Dazu muss der Pfad oder die Referenznummer, unter der die Messwerte gespeichert sind, angegeben werden. In dem VI werden die Spalten der Temperatur und der Luftfeuchtigkeit in einzelne Arrays geschrieben. Die Anzahl der Messwerte wird berechnet. Weil das Array, das ausgegeben wird, nur 10 Reihen haben darf, müssen nun die Punkte ausgerechnet werden, an denen die Temperatur und die Luftfeuchtigkeit ermittelt werden müssen. Es wird davon ausgegangen, dass die Messwerte in gleichmäßigen Abständen aufgenommen wurden.

Um das zu erreichen, wird die Anzahl der Reihen in der Messwertdatei durch 10 geteilt. Das Ergebnis wird dann in einer for-Schleife mit der Schleifeniteration multipliziert. Im Fall, dass es genau 10 Werte in der Datei gibt, ergeben sich 1 Messwert/Pixel. Es kann aber auch sein, dass weniger als 10 Messwerte in der Tabelle aufgezeichnet wurden. Dann ergibt sich Messwerte/Pixel <1. Für den Fall, dass es mehr als 10 Messwerte gibt ist die Zahl der Messwerte/Pixel>1. Es müssen also für den Fall, dass es nicht genau 10 Messwerte in der Datei gibt auch Zwischenschritte in den Messwerten errechnet werden. Gibt es zum Beispiel nur zwei Messwerte sollte der erste Messwert auf dem ersten Pixel angezeigt werden, der zweite Messwert auf dem letzten Pixel. Dazwischen sollten aber auch Werte dargestellt werden.

Um das zu erreichen habe ich mich dafür entscheiden, mit den Messwerten eine Spline Interpolation durchzuführen. Bei einer Spline Interpolation werden jeweils zwei Messwerte n und n+1 mit einem Polynom dritten Grades interpoliert[1]. Das liefert insbesondere bei weniger als 10 Messwerten sehr gute Ergebnisse. Hat aber auch, im Gegensatz zu einer Approximation, den Vorteil, dass man nicht vorher wissen muss, welchen Grad die Ausgleichsgerade haben sollte. Die Y-Werte sind jeweils die Arrays der Temperatur bzw. Luftfeuchtigkeit. Die X-Werte sind die Stelle, an der die Y-Werte stehen (es wird angenommen, dass alle Werte gleichmäßig weit auseinander liegen). Als xi wird das Array der Messwerte/Pixel übergeben. So ergeben sich jeweils 10 Werte für Temperatur und Luftfeuchtigkeit, die am Ende ebenfalls in einem Array dargestellt werden.

Messwerte plotten:

Hier werden die Interpolierten Werte der Temperatur und Luftfeuchtigkeit in einem 10x10 Array durch unterschiedliche Farben dargestellt. Dabei wird der kleinste Wert in der untersten Reihe mit der jeweiligen unteren Farbgrenze dargestellt. Der höchste Wert wird in der obersten Reihe angezeigt und bekommt den Wert der oberen Farbgrenze. Alle dazwischenliegenden Werte werden in dem Bereich dazwischen mit den Farben, die das ColorScaler Sub-VI berechnet, dargestellt. Dies passiert für die Temperatur und für die Luftfeuchtigkeit. Am Ende werden beide Arrays zusammengefügt und dargestellt.

Wavechart erstellen:

In der VI werden die Messwerte für Temperatur und für Luftfeuchtigkeit in eine Form gebracht, dass sie von einem XY Graph, auf dem Bedienpanel der Haupt-VI, angezeigt werden können. Als Eingang des Sub-VIs wird der Pfad oder die Referenznummer der Messwerte benötigt. Die Datei wird nach Strings durchsucht, die in ein Array geschrieben werden. Dabei ist die erste Spalte der Zeitstempel, die Zweite die Temperatur und die Dritte die Luftfeuchtigkeit. Der XY-Graph benötigt als Eingang ein oder mehrere Cluster mit x- und y-Werten. Als y-Werte werden in diesem Fall Zeitstempel genutzt. Um aus

[1] Vgl. Bronstein, Taschenbuch der Mathematik, S.956

dem String der ersten Spalte einen Zeitstempel zu generieren, wird jede Zeile nach einem String in dem Zeitformat ‚%<%c>T ' durchsucht. Jeder Wert wird auto-Indiziert und mit den jeweiligen eindimensionalen Arrays der Temperaturspalte und der Luftfeuchtigkeitsspalte zu zwei Clustern zusammengefasst. Aus diesen beiden Arrays wird wiederum ein Array erstellt. Dieses Array kann dann dargestellt werden.

Dezimal-Array:

In dem VI wird ein Input aus einer Zahl und eines Strings in einen dreidimensionalen Array geschrieben, in der in jeder Array-Ebene ein Zeichen in einer 5x3 Array aus Farben darstellt.

Wenn eine Nummer an das VI übergeben wird, wird sie in einen String umgewandelt. Der übergebene String wird an den String der Nummer gehängt. Wird keine Nummer übergeben steht der übergebene String allein. Jedes Zeichen in dem String wird mit einer Enum-Konstante verglichen, in der alle vorhandenen Zeichen gespeichert sind. Gibt es eine Übereinstimmung mit einem Zeichen wird der jeweilige State aufgerufen in der eine 5x3 Matrix hinterlegt ist. In der Matrix wird jedes Feld, was im Farb-Array in der jeweiligen Farbe dargestellt werden soll, mit einer Eins markiert. Nun wird jede Zelle der Matrix durchgegangen. Steht dort eine Eins wird die Farbe in die Zelle des Arrays geschrieben. Jedes neue Zeichen wird in eine neue Ebene des Arrays geschrieben. Wurde jedes Zeichen des Strings so verarbeitet, wird das Array zurückgegeben.
Weil für jedes Zeichen nur einmal eine Matrix und ein neuer State in der Enum-Konstante „ZahlenEnum" angelegt werden muss, ist es sehr einfach, neue Zeichen hinzuzufügen. Ich habe bisher die für meine Anwendung notwendigen Zeichen [0,1,2,3,4,5,6,7,8,9,:,%,C,°,.] hinzugefügt.

Num_Str_Zusammen:

In dem VI wird eine Nummer und ein String zu einem Array der Größe 5xn, in dem die Zahl und der String in Pixeln der Farbe der Color Box angezeigt werden. Dafür wird mit der Sub-VI Dezimal-Array einen dreidimensionalen Array erstellt. Anschließend wird jede Ebene an das Ende einer separaten Matrix geschrieben. Dabei wird nach jedem Zeichen eine Spalte frei gelassen. Das Resultierende Array wird ausgegeben.

Disp all:

In diesem VI werden entweder die Daten der Temperatur und der Luftfeuchtigkeit oder die Uhrzeit und das Datum auf der LED-Matrix dargestellt. In beiden Fällen sind mehr als 10 Pixel zur Darstellung erforderlich deshalb ist es notwendig, dass die Schrift scrollt. Man kann außerdem die Zeit, die zwischen einem aktualisieren der Matrix gewartet wird, einstellen.

Gibt es Werte für Temperatur und Luftfeuchtigkeit, wird die Laufschrift für beide Werte angezeigt. Sind beide Werte Null, wird davon ausgegangen, dass die Uhrzeit und das Datum angezeigt werden soll. Im Falle der Temperatur und Luftfeuchtigkeit wird mit dem Sub-VI „Num_Str_Zusammen" an den Wert der Temperatur die Einheit ‚°' und an die Luftfeuchtigkeit das Zeichen ‚%' angehängt. Die resultierenden Arrays werden übereinander in zwei unterschiedlichen Farben dargestellt. Im Falle, dass nur die Zeit und das Datum angezeigt werden soll, erwartet das Programm nur Strings. Die Farbe der Zeit mit Rot eingestellt, das Datum mit Grün. Im Programm wird geprüft, wie lang das Array ist und stellt anfangs die Reihen 0-9 auf der LED-Matrix dar. Nachdem die Wartezeit abgelaufen ist und es werden die Reihen 2-11 dargestellt usw. Hierfür wird das Sub-VI „Serielles schreiben" genutzt

Es ist nicht notwendig eine Wartezeit einzustellen, weil es relativ lange dauert bis der Arduino alle Werte empfangen und dargestellt hat.

LauftextTempHum:

In dem VI werden die Werte der Temperatur und Luftfeuchtigkeit auf der LED-Matrix angezeigt. Dazu werden die Sensorwerte über das Sub-VI „Serielles Lesen" ausgelesen und den resultierenden Werten die nötigen Farben über das „Color Scaler" VI zugeordnet. Dafür werden die Standardgrenzen für die Farben und die Zahlen genutzt. An das „Disp all" Sub-VI werden die beiden berechneten Farben und

die Werte der Temperatur und der Luftfeuchtigkeit übergeben. Außerdem wird die gewünschte Intensität übergeben.

Lauftext Uhr:

Hier werden die aktuelle Uhrzeit und das aktuelle Datum auf der LED-Matrix angezeigt. Dazu wird an das Sub-VI die Uhrzeit mit dem Formatstring ‚%I:%M' und das Datum mit ‚%d.%m' übergeben sowie der gewünschte Helligkeit.

Hauptprogramm:

Wie bei einer State-Maschine üblich, wird als Anfangs-State der „Init state" übergeben. Von hier aus laufen die einzelnen Zustände nacheinander ab.

Im „Daten schreiben" State wird das Sub-VI „Daten schreiben" aufgerufen. Dieses VI schreibt sowohl den Pfad als auch die Ref Nummer der Messwerte in ein Schieberegister. Je nach Schalterstellung auf dem Frontpanel ist der nächste Schritt „Laufschrift Temp+Hum" oder „Laufschrift Zeit u Date"

In den States „Laufschrift Temp + Hum" und „Laufschrift Time u Date" werden die jeweiligen Sub-VIs aufgerufen.

Im „Diagramm Plotten" State werden über das „Messwerte Plotten" Sub-VI die Messwerte aus der Datei auf der LED-Matrix dargestellt. Über eine Sequenz-Struktur wird anschließend das Sub-VI „Wavechart erstellen" ausgeführt und das resultierende Array wird auf einem XY Graph dargestellt. In dem State wird außerdem die einstellbare minimale Zeit zwischen Messwerten mit dem Express-VI „Elapsed Time" verglichen. Ist die Zeit noch nicht abgelaufen wird der State „Daten schreiben" übersprungen.

Anwendung des Projektes

Ziel des Projektes war es, mit dem Programm die Luftfeuchtigkeit und die Temperatur über einen abgeschlossenen zeitlichen Rahmen zu messen. Dabei soll man auf der LED-Matrix stets die relevanten Informationen und den bisherigen Trend der aufgezeichneten Verläufe ablesen können.

Anwendungsszenario

Um die Funktion des Programms zu testen, habe ich mich dazu entschieden, die Klimawerte während eines Indoor-Workouts auf dem Fahrrad zu messen. Dabei wird das Fahrrad auf einen Rollentrainer gespannt, mit der der Widerstand am Hinterreifen automatisch eingestellt wird. In dem zu betrachtenden Fall dauert das Workout 90 Minuten, es soll eine durchschnittliche Leistung von 315W erbracht werden (siehe Abbildung 3).

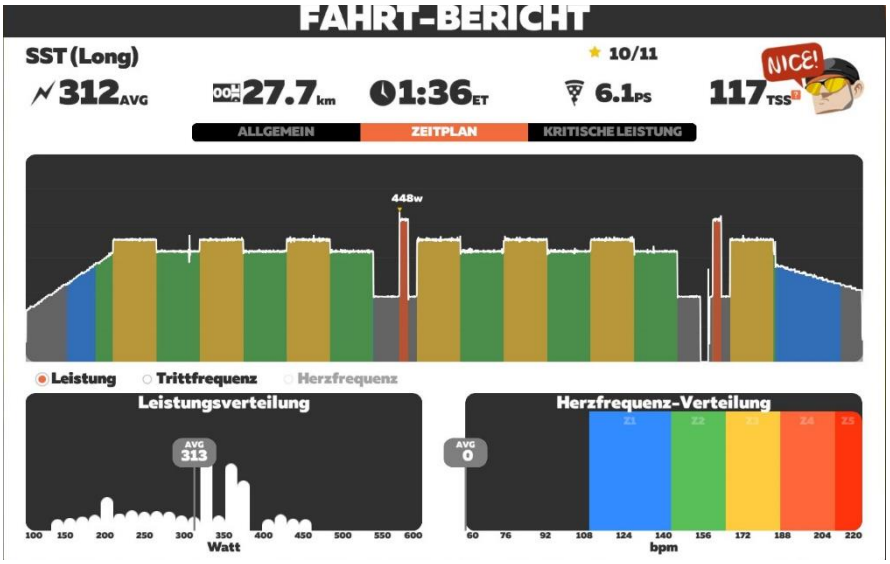

Abbildung 3; Leistungsverlauf des Workouts

Während des Workouts ist das Fenster geöffnet. Außerdem sind zwei Ventilatoren für eine bessere Kühlung angeschaltet.

Es ist zu erwarten, dass die Luftfeuchtigkeit aufgrund von Transpiration nach kurzer Zeit ansteigt. Man kann außerdem erwarten, dass die Raumtemperatur ebenfalls aufgrund der Verlustwärme des Rollentrainers und aufgrund einer Erhöhung der Körpertemperatur steigt.

Durchführung

Das Frontpanel des VIs war während der gesamten Durchführung geöffnet. Aufgrund der absehbaren Länge der Datenaufzeichnung wurde die minimale Zeit zwischen Datenpunkten auf 30 Sekunden gesetzt. Über die Anzeige der Temperatur und der Luftfeuchtigkeit auf dem Frontpanel konnte man die jeweils aktuellen Werte gut ablesen. Es hat sich aber auch die LED-Matrix angeboten. Auf ihr wurde in periodischem Abstand die Temperatur (oben) und die Luftfeuchtigkeit (unten) angezeigt (Abbildung 4). In dem Beispiel beträgt die Temperatur 20°C und die Luftfeuchtigkeit beträgt 63%.

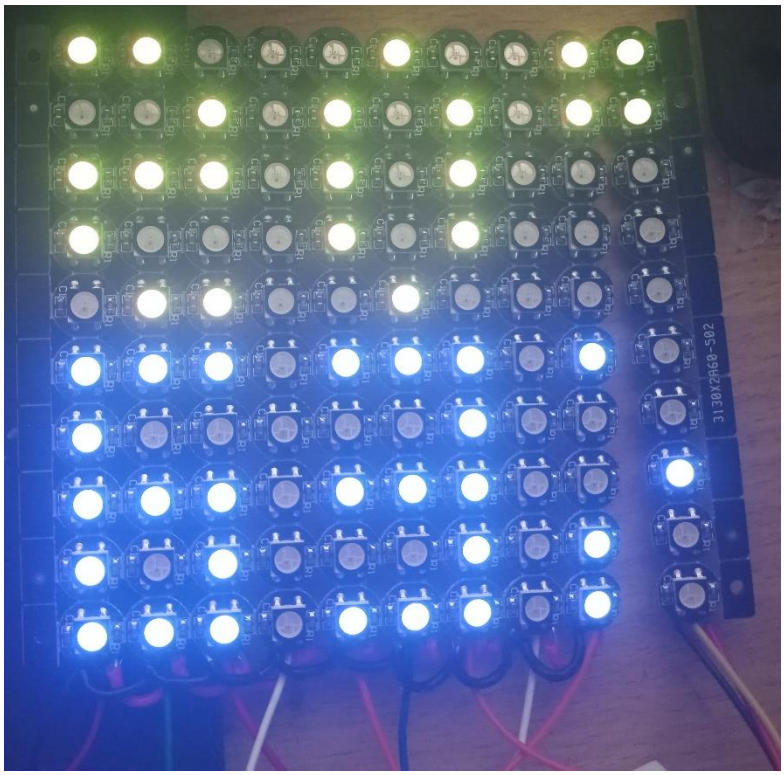

Abbildung 4; Darstellung der Temperatur (gelb) und Luftfeuchtigkeit(blau)

Man hatte ebenfalls eine gute Übersicht über die jeweiligen Tendenzen aufgrund der Darstellung aller bisherigen Messpunkte bzw. der Interpolierten Zwischenschritte:

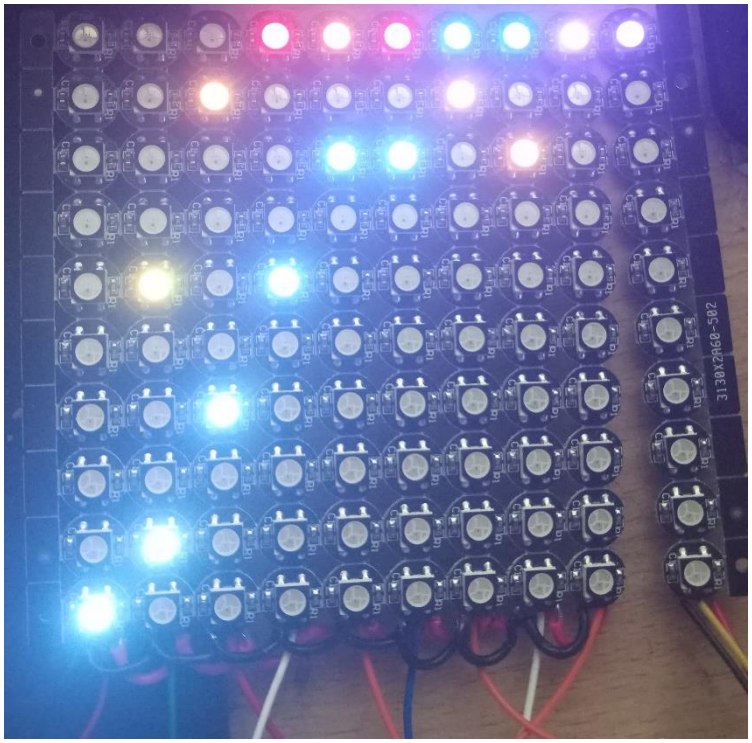

Abbildung 5;Darstellung des Zeitlichen Verlaufes der Daten

Man erkennt, dass die Luftfeuchtigkeit (blau) zu Beginn der Messung ihr Minimum hat. Sie steigt über den Messverlauf von 90 Minuten kontinuierlich an und hat am Ende (letzte Spalte) ihren Maximalwert.

Die Temperatur schwankt über den Verlauf der Messung. Gerade an der Temperatur erkennt man, dass nicht nur die Höhe, in der die Pixel leuchten, eine Aussage über die Temperatur ermöglicht, sondern auch die Stärke der Farbe. Bei der Temperatur ist die maximale Temperatur ein reines Rot. Ist die Farbe eher orange ist die Temperatur geringer. So kann man selbst bei mehreren Werten in der obersten Reihe immer noch eine Aussage darüber treffen, zu welchem Zeitpunkt eine höhere Temperatur gemessen wurde. Diese Messung veranschaulicht ebenfalls was passiert, wenn die Temperatur und die Luftfeuchtigkeit auf dem gleichen Pixel angezeigt werden. In der ersten- und in den letzten beiden Spalten ist jeweils nur eine LED am Leuchten. Gerade in den letzten beiden Spalten erkennt man jedoch, dass sie weder ein reines blau noch ein reines rot ist. Vielmehr ist sie eine Mischung aus rot und blau; also Violett.

Mithilfe dieser Darstellung kann man lediglich einen Trend der Verläufe abschätzen. Man kann keine Aussage über tatsächliche Temperaturen treffen. Wenn man also eine höhere Auflösung und absolute Messwerte in einem zeitlichen Verlauf analysieren möchte, ist es besser, sich den Plot auf dem Frontpanel anzuschauen:

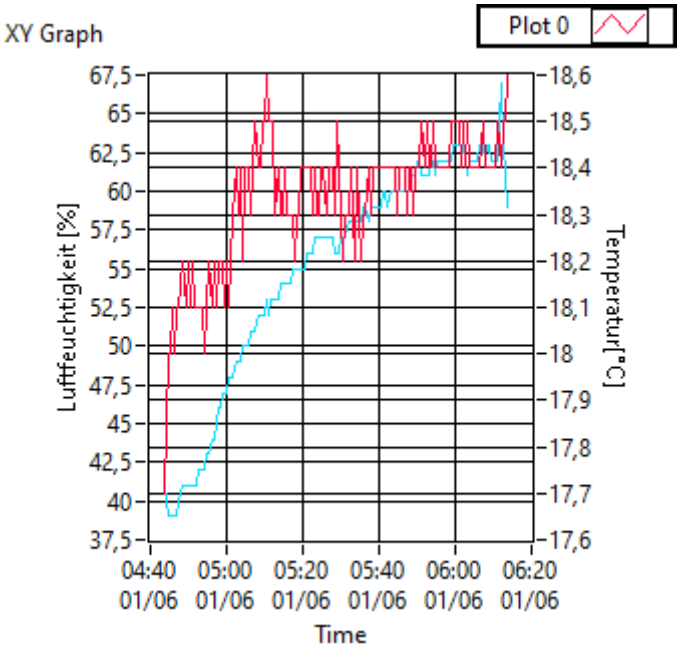

Abbildung 6; XY Chart

Dies ist die Darstellung auf dem Frontpanel. Hier sind die jeweiligen Datenpunkte der Temperatur (rot) und der Luftfeuchtigkeit (blau) nach der Zeit aufgetragen. Es ist außerdem eine Achsenbeschriftung vorhanden. Man erkennt sofort, dass die Luftfeuchtigkeit tatsächlich von einem Wert von ca. 42% um 4:40 auf einen Endwert von ca. 66% um 6:20 steigt. Diesen Verlauf hat die Darstellung auf der LED-Matrix sehr gut widergespiegelt.

Die Temperatur hingegen steigt zwar auch über die Zeit an, die Messwerte befinden sich aber nur zwischen 17,7°C und 18,6°C. Der auf der LED-Matrix dargestellte Trend ist also nicht sehr repräsentativ bzw. irreführend. Durch die Darstellung mit dem minimalen Wert in der untersten Zeile und dem obersten Wert in der obersten Zeile entsteht der Eindruck, dass die Temperaturwerte einen großen Wertbereich überspannen. Außerdem sind die Ausschläge in der Temperatur relativ zum Gesamtverhalten sehr groß. Ein Maximum kann also leicht durch einen fehlerhaften Messwert zustande kommen. Am Ende der 90 Minuten wurden insgesamt 155 Datenpunkte in die Datei „Meswerte220106.txt“ geschrieben.

Fazit zur Funktion

In der Anwendung des Programms ist es wichtig, dass man die Anzeige auf der LED-Matrix nur als tendenziellen Wert betrachtet. Wenn einen die absolute Temperatur oder Luftfeuchtigkeit zu einem

bestimmten Zeitpunkt interessiert, ist die Anzeige auf dem Frontpanel zu benutzen. Um auf einen Blick die aktuelle Temperatur, Luftfeuchtigkeit oder den bisherigen Verlauf zu sehen bietet die LED-Matrix eine sehr gute Möglichkeit. Letztlich sollte man sich vor einer Messung immer die Frage stellen, welche Messwerte zu erwarten sind und welche Darstellungsform für die Fragestellung geeignet ist. In diesem Anwendungsbeispiel war es wichtiger die Trends möglichst schnell und unkompliziert zu erkennen und in dem Punkt erwies sich die Darstellungsform auf der LED-Matrix als sehr angenehm.

Potenzielle Fehler

Producer-Consumer-Entwurfsmuster:

Das Producer-Consumer-Entwurfsmuster stellt sicher, dass mehrere Prozesse nicht gleichzeitig auf Daten zugreifen und unterschiedliche Operationen (lesen/schreiben) ausführen. Das könnte zu widersprüchlichen Daten führen[2].

In dem Projekt gibt es zwei mögliche stellen, an denen es zu Problemen kommen könnte. Zwei oder mehr VIs könnten gleichzeitig auf die Serielle Schnittstelle zugreifen und so den Datenaustausch behindern oder die gesendeten Daten unbrauchbar machen. Außerdem könnten zwei Programmteile auf die Messwertdatei zugreifen.

Weil das Programm als State-Maschine ausgeführt ist, es also keine zwei parallellaufenden Schleifen gibt, muss darauf geachtet werden, dass in keinem State versucht wird eine serielle Verbindung doppelt zu öffnen und nachdem der Datenaustausch abgeschlossen ist, die Verbindung wieder zu schließen. In dem Programm kommt es zu keinem Zeitpunkt dazu, dass gleichzeitig zwei serielle Verbindungen aufgebaut werden könnten.

Es ist außerdem darauf zu achten, dass jeweils nur ein Programm eine Operation mit der Messwerte-Datei ausführt. In dem State „Messwerte schreiben“ wird nur eine Operation mit der Datei ausgeführt. In dem State „Messwerte Plotten“ könnte es hingegen zu einem gleichzeitigen Zugriff auf die Messwerte-Datei kommen. Darum kommt die Sequenz-Struktur zum Einsatz. So wird erst die erste Operation (Messwerte auslesen und an die Matrix schreiben) ausgeführt. Erst wenn der Prozess abgeschlossen ist, wird wieder auf die Datei zugegriffen, um die Messwerte in dem Graphen auf dem Front-Panel darzustellen.

Stromversorgung der Neopixel:

In dem Datenblatt der WS2812 Neopixel wird pro Farbe ein Strom von 20mA/LED angegeben[3]. Wird auf der LED also die Farbe Weiß angezeigt, beträgt der Strom 60mA. In meiner Messung ergibt sich ein Grundstrom von 88mA, ohne dass eine LED eine Farbe darstellt. Wird nun eine der LEDs auf die Farbe Weiß bei maximaler Helligkeit gestellt beträgt der Gesamtstrom 110mA. Wenn kein externes Netzteil zur Stromversorgung der Neopixel genutzt wird, muss der Strom vom Arduino und seine USB-Verbindung mit dem PC kommen. Der Arduino hat einen maximalen Outputstrom von 800mA[4]. Ist der Arduino allerdings nur an einen USB 2.0 Port angeschlossen beträgt der maximale Strom 500mA (vgl. USB 2.0 Spezifikation, S.178)[5]. Wenn also kein zusätzliches Netzteil genutzt wird, kann es zu Problemen kommen, wenn zu viele LEDs zu hell leuchten.

---

[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

Um experimentell festzustellen wie viel Strom maximal benötigt wird, wurde die Stromversorgung mit einem Elegoo Power MB V2 realisiert. Dabei wurden alle 100 LEDs auf den RGB-Wert (255,255,255) gesetzt. Dieser Wert wurde dann mit der Helligkeit (0-1) multipliziert.

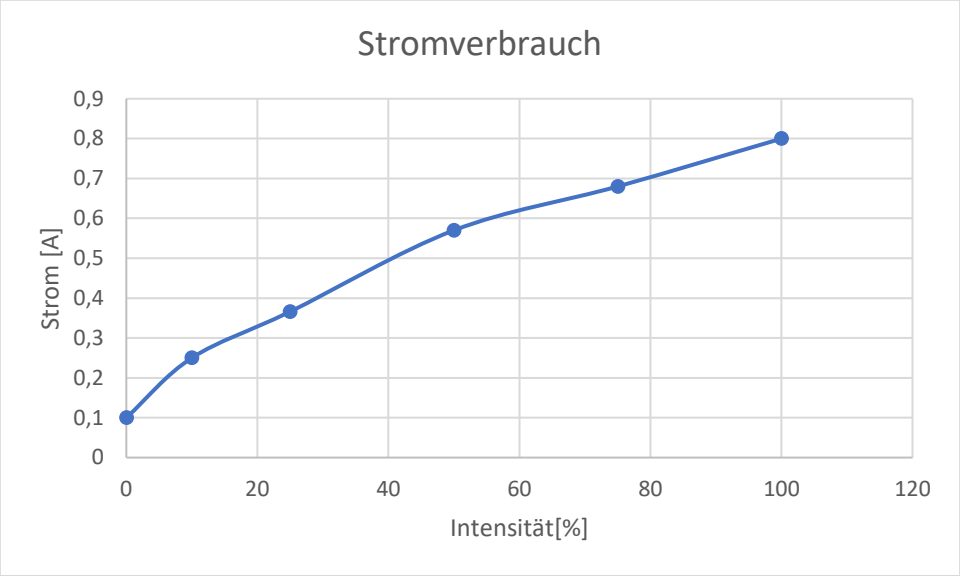

Abbildung 7; Stromverbrauch der Neopixel

Man erkennt, dass bei maximaler Helligkeit ein Strom von 800mA benötigt wird. Diesen Strom könnte ein USB 2.0 Port nicht mehr bereitstellen und es kann zu falsch angezeigten Farben kommen. In dem Projekt kommt es aktuell nicht dazu, dass so viele LEDs gleichzeitig angeschaltet sind. Außerdem ist die Helligkeit standartmäßig auf 10% eingestellt. Wenn man jedoch neue anziehbare Seiten implementiert, darf man den Stromverbrauch nicht außer Acht lassen.

Darstellung der Kurven auf der Neopixel Matrix

Wenn zwei Messpunkte auf einem Pixel angezeigt werden sollen, werden die RGB-Werte der beiden Farben, die dargestellt werden sollen, addiert. Im Normalfall ist die Helligkeit in dem Frontpanel auf einen Wert eingestellt, der deutlich kleiner als 50% ist. So kann der maximal dargestellte RGB-Wert nur (127,127,127) sein. Werden nun im Worst-Case zwei solcher Werte addiert, entsteht trotzdem noch eine darstellbare Farbe, die auf der Matrix angezeigt werden kann. Ist die Helligkeit auf einen Wert eingestellt, der größer als 50% ist, kann es zu Teilen des RGB-Wertes kommen, die größer als 255 sind. Ist das der Fall wird die jeweilige Farbe verändert angezeigt. So wird ein RGB-Wert von (256,256,256) als (1,1,1) dargestellt. Das kann insbesondere beim „Mischen“ zu unerwünschten Farben führen.

Potenzielle Erweiterungen

Aufgrund der Implementierung der Laufschrift ist es möglich diverse Messwerte oder Texte auf der LED-Matrix anzeigen zu lassen. Das gleiche gilt selbstverständlich für das Front Panel. Über den Arduino kann außerdem eine große Anzahl an Sensoren angeschlossen werden. So könnte man zum Beispiel nicht die Temperatur und Luftfeuchtigkeitsdaten auswerten, sondern die Feuchtigkeit von Böden. Die LED-Matrix könnte so anzeigen, wann ein bestimmtes Limit unterschritten wird.