LED Echtzeituhr mit alternierender Temperatur und Luftfeuchteanzeige (Teil 2)

Hallo und willkommen zu einem neuen Teil rund um unser 4 Bit Digital Tube LED Display. Im heutigen zweiten Teil der Reihe erweitern wir die Anzeigeparameter des Displays um Temperatur und Luftfeuchte. Dabei wechselt die Anzeige in frei im Code einstellbaren Intervallen zwischen Uhrzeit Temperatur in Celsius und relativer Luftfeuchtigkeit in Prozent. Die Uhrzeit ist trotzdem weiterhin jederzeit bequem mit zwei Tastern einstellbar. Auch in diesem Teil verzichten wir noch auf eine externe RTC Uhr.

Als Temperatur und Feuchtesensor verwenden wir den bekannten und zuverlässigen DHT 22. Da dieser jedoch mit 3,3 Volt Logiklevel arbeitet, während der Nano mit 5,0 Volt Logiklevel arbeitet, verwenden wir als Logiklevelshifter den sehr guten uns universell einsatzbaren Baustein TXS0108E.

Als Teileliste für unser heutiges Projekt benötigen wir insgesamt also:

Anzahl	Beschreibung	Anmerkung
1	DHT 22	
	DHT 11	Alternativ zu DHT 22
2	KY-004 Button Module	
1	Nano V3	
1	4 Digit 7 Segment Display (TM1637)	
1	MB102 Netzteil Adapter	Für Breadboardaufbau
1	Logic Level Converter TXS0108E	

Anstelle des zuerst gelisteten DHT 22 Sensors auch der preisgünstigere DHT 11 Sensor durch einfaches Anpassen der Zeile „DHTTYPE DHT22“verwendet werden. Der DHT 11 ist jedoch nicht so Messgenau wie der DHT 22.

Hier finden sich die Unterschiede der beiden Sensoren auch noch mal zum Nachlesen.

Wenn der DHT 11 Sensor verwendet werden soll, muss die Zeile

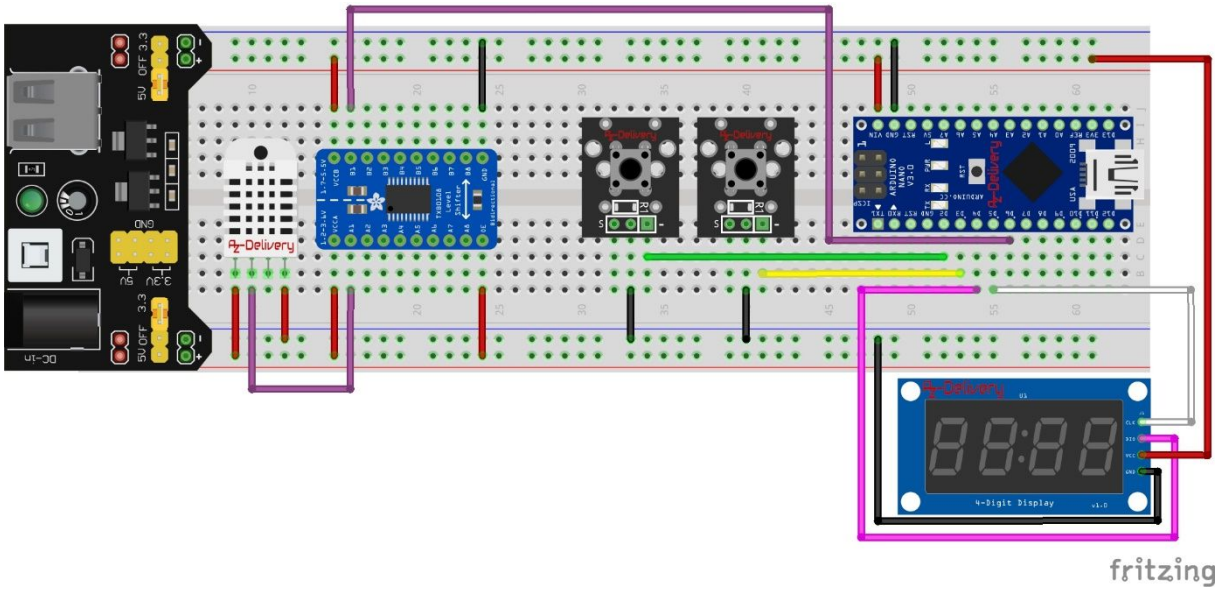
```
#define DHTTYPE DHT22
```

in

```
#define DHTTYPE DHT11
```

geändert werden. Weitere Änderungen sind nicht notwendig.

Wir verdrahten die Teile wie auf folgendem Bild ersichtlich miteinander:



Bevor wir nun zur eigentlichen Programmierung schreiten können, müssen wir uns einige Bibliotheken in unsere IDE laden.

Da wäre zunächst die aus dem ersten Teil bekannte Bibliothek TM1637 Driver by AKJ Bibliothek zur Ansteuerung des TM1637. Für den DHT22 Sensor brauche wir darüber gehend hinaus noch die generalisierte „Adafruit Unified Sensor Libary“ : als auch die eigentliche DHT Sensor Library. Beide Bibliotheken bauen aufeinander auf und müssen zu unseren Bibliotheken der IDE hinzugefügt werden, da sie von unserem Projekt benötigt werden.

Nach Hinzufügen der Bibliotheken und evtl. Anpassungen der Parameter im Code, laden wir folgenden Code auf unseren ESP hoch:

```
// Code by Tobias Kuch 2019, Licesed unter GPL 3.0

#include <TM1637.h>
#include "DHT.h"    // REQUIRES the following Arduino libraries:
                    //- DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
                    //- Adafruit Unified Sensor Lib:
https://github.com/adafruit/Adafruit_Sensor
// Instantiation and pins configurations
// Pin 4 - > DIO
// Pin 5 - > CLK
TM1637 tm1637(4, 5);
#define BUTTON_MINUTEUP_PIN   2    // Digital IO pin connected to the button.
This will be
                    // driven with a pull-up resistor so the switch should
                    // pull the pin to ground momentarily.  On a high -> low
                    // transition the button press logic will execute.
                    // Used for Setting the Clock Time

#define BUTTON_HOURUP_PIN   3    // Digital IO pin connected to the button.
This will be
                    // driven with a pull-up resistor so the switch should
                    // pull the pin to ground momentarily.  On a high -> low
                    // transition the button press logic will execute.
                    // Used for Setting the Clock Time
//DHT Konfiguration
#define DHTPIN 6                        // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22                       // DHT 22  (AM2302), AM2321

DHT dht(DHTPIN, DHTTYPE); // DHT Sensor Instanz initalisieren

struct DHTSensorData
  {
    byte Humidity = 0 ;      // Luftfeuchtigkeitssensordaten in Prozent
    byte Temperature = 0;
    bool DataValid  = false;
    bool SensorEnabled  = false;
  };

DHTSensorData DHTMeasure;

// interrupt Control
bool SecInterruptOccured = true;
bool A60telSecInterruptOccured = true;
byte A60telSeconds24 = 0;



// Clock Variables
byte Seconds24;
byte Minutes24 ;
byte Hours24;
byte Displayalternation = 0;
```

```
bool DisableSecondDisplay = false;
bool MinSetQuickTime = false;
bool HourSetQuickTime = false;
bool ButtonDPress = false;
bool ButtonEPress = false;

//Interrupt Routines
ISR(TIMER1_COMPA_vect)
{
  A60telSeconds24++;
  if ((A60telSeconds24 > 59) and !(MinSetQuickTime))
    {
       A60telSeconds24 = 0;
      //Calculate Time 24 Stunden Format
      SecInterruptOccured = true;
      Seconds24++;
      if (Seconds24 > 59)
        {
          Seconds24 = 0;
          Minutes24++;
        }
      if (Minutes24 > 59)
        {
          Minutes24 = 0;
          Hours24++;
        }
      if (Hours24 > 23)
        {
          Hours24 = 0;
        }
    }
    if  (MinSetQuickTime)
    {
       A60telSeconds24 = 0;
      //Calculate Time 24 h Format
      SecInterruptOccured = true;
      Seconds24++;
      if (Seconds24 > 59)
        {
          Seconds24 = 0;
          Minutes24++;
        }
      if (Minutes24 > 59)
        {
          Minutes24 = 0;
          Hours24++;
        }
      if (Hours24 > 23)
        {
          Hours24 = 0;
        }
```

```
    }
  TCNT1 = 0;      // Register mit 0 initialisieren
if  (HourSetQuickTime)
    {
     OCR1A =  200;
    } else
    {
     OCR1A =  33353;      // Preload Output Compare Register
    }
  A60telSecInterruptOccured = true;
}

//Interrupts ende
void CheckConfigButtons ()    // InterruptRoutine
{
bool PressedZ;
PressedZ= digitalRead(BUTTON_MINUTEUP_PIN);
if ((PressedZ == LOW) and (ButtonDPress == false))
   {
    ButtonDPress = true;
    delay(100);
    Minutes24++;
    Seconds24 = 0;  // Reset Seconds to zero to avoid Randomly time
    DisableSecondDisplay = true ;   // Disable Seconds While Clock Set
    MinSetQuickTime = true; //Enable Quick Tmime Passby
   }
if ((PressedZ == HIGH) and (ButtonDPress == true))
   {
   ButtonDPress = false;
   delay(100);
   DisableSecondDisplay = false ;   // Enable Seconds While Clock Set
   MinSetQuickTime = false;
   Seconds24 = 0;  // Reset Seconds to zero to avoid Randomly time
   A60telSeconds24 = 0;

   }
PressedZ= digitalRead(BUTTON_HOURUP_PIN);
if ((PressedZ == LOW) and (ButtonEPress == false))
   {
      ButtonEPress = true;
      delay(100);
      DisableSecondDisplay = true ;   // Disable Seconds While Clock Set
      MinSetQuickTime = true; //Enable Quick Tmime Passby
      HourSetQuickTime = true;
   }
if ((PressedZ == HIGH) and (ButtonEPress == true))
   {
      noInterrupts();   // deactivate Interrupts
      ButtonEPress = false;
      delay(100);
      Minutes24++;
```

```
      DisableSecondDisplay = false ;   // Enable Seconds While Clock Set
      MinSetQuickTime = false; //Enable Quick Tmime Passby
      HourSetQuickTime = false;
      Seconds24 = 0;  // Reset Seconds to zero to avoid Randomly time
      A60telSeconds24 = 0;
      interrupts();   // enable all Interrupts
    }
}

void setup()
{
    tm1637.init();
    Serial.begin(9600);
    tm1637.setBrightness(8); // Highest Brightness
    pinMode(BUTTON_MINUTEUP_PIN, INPUT_PULLUP);
    pinMode(BUTTON_HOURUP_PIN, INPUT_PULLUP);
    digitalWrite(LED_BUILTIN, LOW);
    noInterrupts();
    TCCR1A = 0x00;
    TCCR1B =  0x02;
    TCNT1 = 0;      // Register mit 0 initialisieren
    OCR1A =  33353;      // Output Compare Register vorbelegen
    TIMSK1 |= (1 << OCIE1A);  // Timer Compare Interrupt aktivieren
    interrupts();
    Seconds24 = 1;
    Minutes24 = 1;
    Hours24 = 0;
    dht.begin();
}


void DisplayHumityOnTM1637()
{
byte Humidity = dht.readHumidity();
byte n = (Humidity / 10) % 10; //zehner
byte m = Humidity % 10; // einer
if (Humidity < 100)
  {
  tm1637.display(0,n); // Digit 1
  tm1637.display(1,m); // Digit 2
  tm1637.display(2,104); // Clear Digit
  } else
  {
  tm1637.display(0,103); // - Sign
  tm1637.display(1,103); // - Sign
  tm1637.display(2,103); // - Sign
  }

tm1637.display(3,160); // Special Character
}
```

```
void DisplayTempOnLedTM1637()
{
int Temperature = dht.readTemperature(false);   // Read temperature as Celsius
(isFahrenheit = true)
byte n = (Temperature / 10) % 10; //zehner
byte m = Temperature % 10; // einer
if (Temperature < 0)
  {
  tm1637.display(0,103); // - Sign
  tm1637.display(1,n); // Digit 1
  tm1637.display(2,m); // Digit 2
  } else if (Temperature < 99)
  {
  tm1637.display(0,104); // Clear Digit
  tm1637.display(1,n); // Digit 1
  tm1637.display(2,m); // Digit 2
  } else
  {
  tm1637.display(0,103); // - Sign
  tm1637.display(1,103); // - Sign
  tm1637.display(2,103); // - Sign
  }
tm1637.display(3,99); // C Character
}

void DisplayClockOnLedTM1637()
{
  if (!(DisableSecondDisplay)) {tm1637.switchColon();}
  tm1637.dispNumber(Minutes24 + Hours24 * 100);
}

void loop()
{
  bool PressedC;
  if (A60telSecInterruptOccured)
    {
      A60telSecInterruptOccured = false;
    }
  if (SecInterruptOccured)
    {
        SecInterruptOccured = false;
        if (!DisableSecondDisplay) {Displayalternation ++;}
        if (DisableSecondDisplay) {Displayalternation = 16;}
        if ((Displayalternation < 8) & (!DisableSecondDisplay))
          {
          DisplayTempOnLedTM1637();
          } else if ((Displayalternation < 15)  & (!DisableSecondDisplay))
          {
           DisplayHumityOnTM1637();
          } else if ((Displayalternation < 35) | (DisableSecondDisplay))
          {
```

```
          DisplayClockOnLedTM1637();
        } else
        {
         Displayalternation = 0;
        }
     }
   CheckConfigButtons();
}
```

Im nachfolgenden Teil werden wir unserer Uhr eine RTC Uhr als Zeitsynchronisation und als Buffer für einen Stromausfall spendieren.

Ich wünsche viel Spaß und bis zum nächsten Mal.