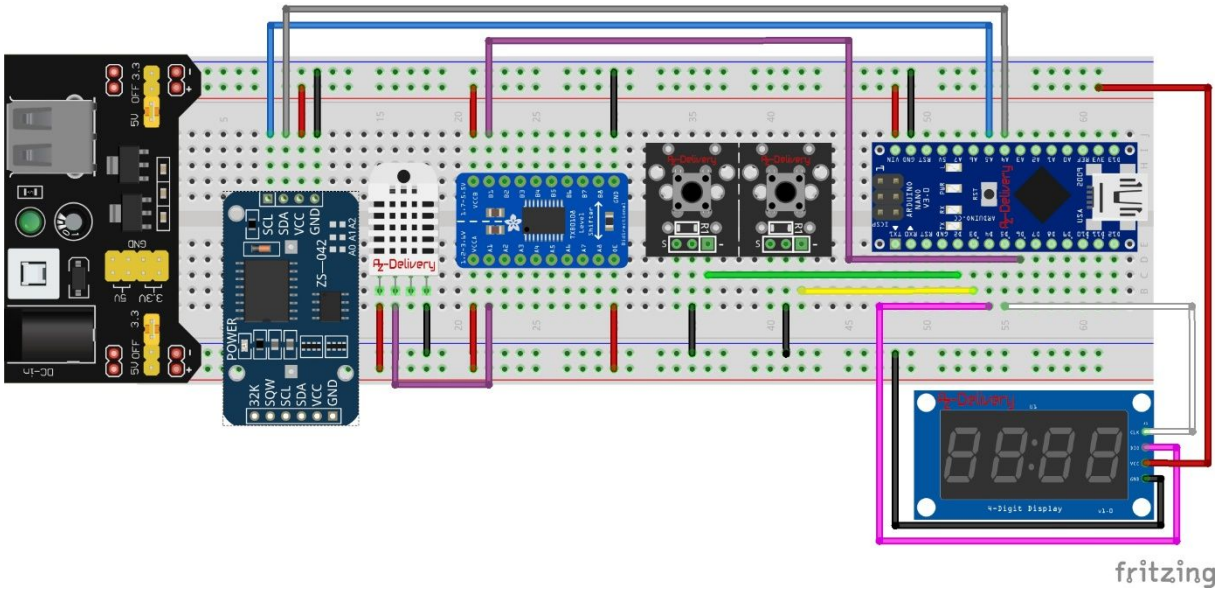
LED Echtzeituhr mit RTC Modul, alternierender Temperatur und Luftfeuchteanzeige (Teil 3)

Hallo und willkommen zum dritten Teil rund um unser 4 Bit Digital Tube LED Display. Im heutigen Teil der Reihe beseitigen wir nicht nur die im ersten Teil der Reihe angesprochene Schwachstelle, dass die Uhr nach jedem Neustart oder nach jedem Spannungsausfall neu eingestellt werden muss, sondern fügen darüber hinaus noch eine serielle Schnittstelle hinzu, über die die Uhr ebenfalls einstellbar wird ! Zunächst jedoch erst einmal zu dem Ausbau der Hardware. Wir spendieren unserer Uhr ein RTC Echtzeitmodul mit Batteriepufferung. Dieses Modul kann über I2C angesprochen werden und jederzeit abgefragt, oder auch gesetzt werden. Die Abfrage der Uhrzeit nutzen wir sowohl bei einem Neustart der Uhr, als auch periodisch zum Abgleich zwischen der internen und RTC Uhr. Gesetzt wird die Uhrzeit durch Einstellen der Uhrzeit mittels Taster ODER serieller Schnittstelle.

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	
1	Real Time Clock RTC DS3231	I2C Echtzeituhr für Arduino

Die Verdrahtung der Teile erfolgt wie folgt:



Nun können wir den erweiterten Code hochladen:

```
// Code by Tobias Kuch 2019, License 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
#include <Wire.h>
// 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
#define DS3231_I2C_ADDRESS 0x68
#define MaxInputBufferSize 5 // maximal 255 Zeichen anpassen an vlcdr

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;
  };

//Serial Input Handling
char TBuffer;
char Cbuffer[MaxInputBufferSize+1];       //USB Code Input Buffer
String Sbuffer = "";                      //USB String Input Buffer
int value;                            //USB Nummeric Input Buffer
byte Ccount { 0 };                        //Number received Chars
byte Inptype = 0;
boolean StrInput = false;
boolean NumberInput = false;
boolean DataInput = false;
boolean EnterInput = false;
byte MenueSelection = 0;
byte MnuState = 0;            // Maximale Menuetiefe 255 icl Sub

// 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;      // Output Compare Register vorbelegen
   }
  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;
  setDS3231time( Seconds24,Minutes24,Hours24,1,24,6,77);
  }
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
    setDS3231time( Seconds24,Minutes24,Hours24,1,24,6,77);
  }
}

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();
    Wire.begin();
    Serial.flush();
    readDS3231time(&Seconds24,&Minutes24,&Hours24);

}

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(3,56);
}

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);
}

byte decToBcd(byte val)
{
  return( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
  return( (val/16*10) + (val%16) );
}

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte
dayOfMonth, byte month, byte year)
{
  // sets time and date data to DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // set next input to start at the seconds register
  delay(10);
  Wire.write(decToBcd(second)); // set seconds
  delay(10);
  Wire.write(decToBcd(minute)); // set minutes
  delay(10);
  Wire.write(decToBcd(hour)); // set hours
  delay(10);
  Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
  delay(10);
  Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
  delay(10);
  Wire.write(decToBcd(month)); // set month
```

```
  delay(10);
  Wire.write(decToBcd(year)); // set year (0 to 99)
  delay(10);
  Wire.endTransmission();
}

void readDS3231time(byte *second,byte *minute,byte *hour)
{
  byte dummy;
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // set DS3231 register pointer to 00h
  Wire.endTransmission();
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
  // request seven bytes of data from DS3231 starting from register 00h
  while(Wire.available())    // slave may send less than requested
  {
  *second = bcdToDec(Wire.read() & 0x7f);
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);
  dummy = bcdToDec(Wire.read());
  dummy = bcdToDec(Wire.read());
  dummy = bcdToDec(Wire.read());
  dummy = bcdToDec(Wire.read());
  }
}

void ScheduledTasks ()

{
   if ((Hours24 == 6) and (Minutes24 == 00) and (Seconds24== 00) )
   {
     readDS3231time(&Seconds24,&Minutes24,&Hours24);
   }
   if ((Hours24 == 12) and (Minutes24 == 00) and (Seconds24== 00) )
   {
     readDS3231time(&Seconds24,&Minutes24,&Hours24);
   }
   if ((Hours24 == 18) and (Minutes24 == 00) and (Seconds24== 00) )
   {
     readDS3231time(&Seconds24,&Minutes24,&Hours24);
   }
   if ((Hours24 == 0) and (Minutes24 == 00) and (Seconds24== 00) )
   {
     readDS3231time(&Seconds24,&Minutes24,&Hours24);
   }
}


//Serial Command Interpreter Functions -------------------------------

void ClearCBuffer ()
```

```
{
 for (byte a= 0; MaxInputBufferSize -1;a++)
  Cbuffer[a] = 0;
}

boolean CheckforserialEvent()
{
  while (Serial.available()) {
     // get the new byte:
     TBuffer = Serial.read();
     if (TBuffer > 9 && TBuffer < 14)
       {
        Cbuffer[Ccount] = 0;
        TBuffer =0;
        Serial.print(char(13));
        Serial.flush();
        Serial.println("");
        Sbuffer = "";
        value = 0;
        EnterInput = true;
       return true;
     } else if (TBuffer > 47 && TBuffer <58 )
      {
       if ( Ccount < MaxInputBufferSize)
          {
            Cbuffer[Ccount] = TBuffer;
            Ccount++;

          } else {Serial.print("#"); }

      //Number Input detected
      NumberInput = true;
      }
      else if (TBuffer > 64 && TBuffer < 123 )
      {
       if ( Ccount < MaxInputBufferSize)
          {
            Cbuffer[Ccount] = TBuffer;
            Ccount++;

            Serial.print(char(TBuffer));
            Serial.flush();
          }
      //Character Char Input detected
      StrInput = true;
     }
    else if ( (TBuffer == 127 )  |  (TBuffer == 8 ) )
        {

        if ( Ccount > 0)
```

```
        {
          Ccount--;
          Cbuffer[Ccount] = 0;

          Serial.print("-");
          Serial.flush();

        }
      }
    else
      {

       if ( Ccount < MaxInputBufferSize)
        {
          Cbuffer[Ccount] = TBuffer;
          Ccount++;

          Serial.print(char(TBuffer));
          Serial.flush();
     //Data Input detected
     DataInput = true;
     }

    return false;
    }
  return false;
  }
}

byte SerInputHandler()
{
byte result = 0;
int c;
int d;
int a;
int b;
result = 0;
if (CheckforserialEvent())
   {
    if ((NumberInput) and not (DataInput)and not (StrInput))     //Numbers only
      {
      Sbuffer = "";
      value = 0;
      StrInput = false;
      NumberInput = false;
      DataInput = false;
      EnterInput = false;
      a = 0;
      b = 0;
      c = 0;
      d = 0;
```

```
      Sbuffer = Cbuffer; // Zahl wird AUCH ! in SBUFFER übernommen, falls
benötigt.
      if (Ccount == 1) { value  = Cbuffer[0]- 48 ; }
      if (Ccount == 2) {
        a = Cbuffer[0] - 48 ;
        a = a * 10;
        b = Cbuffer[1] - 48 ;
        value = a + b;
        }
      if (Ccount == 3) {
        a = Cbuffer[0] - 48 ;
        a = a * 100;
        b = Cbuffer[1] - 48 ;
        b = b * 10;
        c = Cbuffer[2] - 48 ;
        value = a + b + c;
        }
      if (Ccount == 4) {
        a = Cbuffer[0] - 48 ;
        a = a * 1000;
        b = Cbuffer[1] - 48 ;
        b = b * 100;
        c = Cbuffer[2] - 48 ;
        c = c * 10;
        d = Cbuffer[3] - 48 ;
        value = a + b + c + d;
        }
      if (Ccount >= 5)
         {
          Sbuffer = "";
          value = 0;
          Sbuffer = Cbuffer;
          ClearCBuffer;
          result = 2;
         } else
         {
          ClearCBuffer;
          Ccount = 0;
          result = 1;                                    //Number Returncode
          NumberInput = false;
          StrInput = false;
          DataInput = false;
          EnterInput = false;
          Ccount = 0;
          return result;
          }
      }
    if ((StrInput) and not (DataInput))                  //String Input only
      {
      Sbuffer = "";
      Sbuffer = Cbuffer;
```

```
      value = 0;
      StrInput = false;
      NumberInput = false;
      DataInput = false;
      EnterInput = false;
      Ccount = 0;
      ClearCBuffer;
      result = 2;                                    //Number Returncode
      }
     if (DataInput) {
      Sbuffer = "";
      Sbuffer = Cbuffer;
      value = 0;
      StrInput = false;
      NumberInput = false;
      DataInput = false;
      EnterInput = false;

      Ccount = 0;
      ClearCBuffer;
      result = 3;                                  //Number Returncode
      }
      if ((EnterInput) and not (StrInput) and not (NumberInput) and not (DataInput))
      {
      Sbuffer = "";
      value = 0;
      Ccount = 0;
      ClearCBuffer;
      result = 4;                                  //Number Returncode
      }

   NumberInput = false;
   StrInput = false;
   DataInput = false;
   EnterInput = false;
   Ccount = 0;
   return result;

  }
return result;
   //End CheckforSerialEvent
}

void SerialcommandProcessor()
{
int a;
Inptype = 0;
Inptype = SerInputHandler();
// 0 keine Rückgabe
// 1 Nummer
// 2 String
```

```
// 3 Data

if (Inptype > 0)

  {

  MenueSelection = 0;

  if ((MnuState < 2) && (Inptype == 2)) {Sbuffer.toUpperCase(); } // For Easy
Entering Commands

  if ((Sbuffer == "T") && (MnuState == 0) && (Inptype == 2))   { MenueSelection =
1;}
  if ((Sbuffer == "C")&& (MnuState == 0) && (Inptype == 2))       { MenueSelection
= 2;}
  if ((Sbuffer == "B") && (MnuState == 0) && (Inptype == 2))       { MenueSelection
= 3;}
  if ((Sbuffer == "F") && (MnuState == 0) && (Inptype == 2))       { MenueSelection
= 4;}

  if ((MnuState == 2)  && (Inptype == 1))                    { MenueSelection = 8;}
  if (MnuState == 3)                                  { MenueSelection = 9;}
  if (MnuState == 4)                                  { MenueSelection = 10;}
  //Display  Selected Content
  if (MnuState == 9)                                  { MenueSelection = 20;} // Color
Set
  if (MnuState == 10)                                  { MenueSelection = 21;} // Time
Set
  if (MnuState == 11)                                  { MenueSelection = 24;} // Time
Set
  if (MnuState == 12)                                  { MenueSelection = 25;} // Time
Set
  if (MnuState == 13)                                  { MenueSelection = 27;} //
Background Set
  if (MnuState == 14)                                  { MenueSelection = 29;} //
ClockFace Set

  switch (MenueSelection)
   {
     case 1:
     {
     Serial.println("System Time: " + String (Hours24) + ":"+ String (Minutes24) +
":"+ String (Seconds24) );

     Serial.println("Hour: (0-23)");
     MnuState = 12;
     value = 0;
     Sbuffer = "";
     break;
     }
     case 20:
```

```
        {
        value = 0;
        MnuState = 0;
        Sbuffer = "";
        break;
        }
        case 21:
        {
        if ((value >= 0) & (value < 60))
        {
        Seconds24 = value;
        A60telSeconds24 = 0;
        Serial.println("Seconds " + String (value) + " set.");
        Serial.println("Updated new Time: " + String (Hours24) + ":"+ String
(Minutes24) + ":"+ String (Seconds24) );
        MnuState = 0;
        setDS3231time( Seconds24,Minutes24,Hours24,1,24,6,77);
        delay(100);
        } else
        {
        readDS3231time(&Seconds24,&Minutes24,&Hours24);
        value = 0;
        Sbuffer = "";
        MnuState = 0;
        Serial.println("Value out of Range.");
        }
        value = 0;
        MnuState = 0;
        Sbuffer = "";
        break;
        }
        case 24:
        {
         if ((value >= 0) & (value < 60))
        {
        Minutes24 = value;

        Serial.println("Minutes " + String (value) + " set.");
        MnuState = 10;
        Serial.println("Seconds: (0-60)");
        } else
        {
        readDS3231time(&Seconds24,&Minutes24,&Hours24);
        value = 0;
        Sbuffer = "";
        Serial.println("Value out of Range.");
        MnuState = 0;
        }
        value = 0;
        Sbuffer = "";
        break;
```

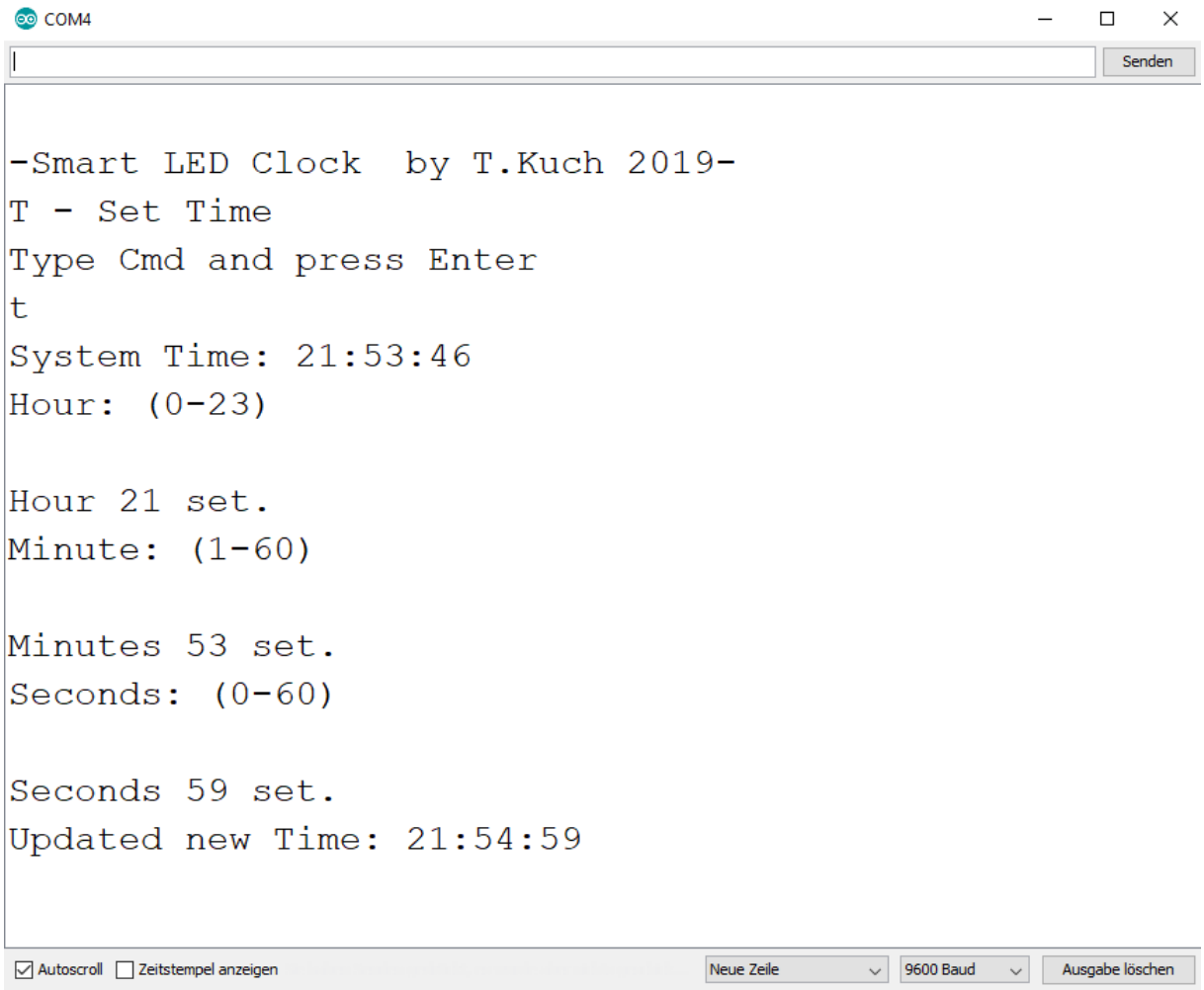
```
      }
       case 25:
      {
       if ((value >= 0) & (value < 24))
      {
      Hours24 = value;
      Serial.println("Hour " + String (value) + " set.");
      MnuState = 11;
      Serial.println("Minute: (1-60)");
      } else
      {
      readDS3231time(&Seconds24,&Minutes24,&Hours24);
      value = 0;
      Sbuffer = "";
      Serial.println("Value out of Range.");
      }
      value = 0;
      Sbuffer = "";
      break;
      }


      default:
      {
        Serial.println("-Smart LED Clock  by T.Kuch 2019-");
        Serial.println("T - Set Time");
        Serial.println("Type Cmd and press Enter");
        Serial.flush();
        MnuState = 0;
        value = 0;
        Sbuffer = "";
      }
    }
   } // Eingabe erkannt
}


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;
        }
       ScheduledTasks();
     }
   CheckConfigButtons();
   SerialcommandProcessor();
}
```

Fertig ! Wenn bis jetzt alles funktioniert hat, können wir uns nun über die serielle Schnittstelle mit 9600 verbinden, und erhalten folgende Menüstrucktur:



Ich wünsche viel Spaß beim Nachbauen und bis zum nächsten Teil der Reihe.