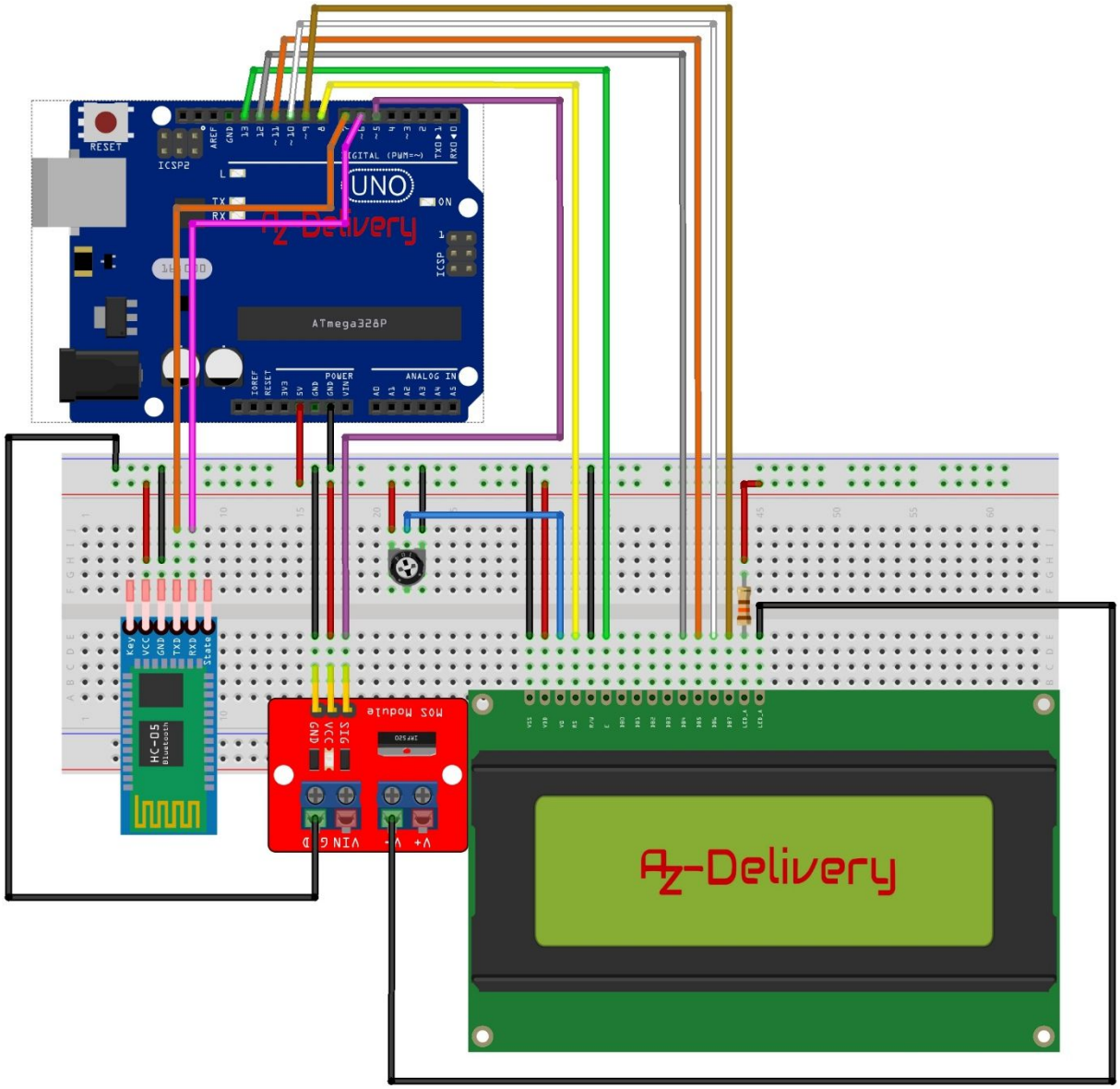
Willkommen zu dem dritten Teil der Bluetooth Display Reihe. Heute bekommt unser Display neben einem obligatorischen weiteren Bedienermenüpunkt eine kleine Erweiterung der Hardware spendiert! Über diese neue Hardwareerweiterung kann die Hintergrund Helligkeit unseres Displays per Software gesteuert und konfiguriert werden!

Dazu verwenden wir den Port 5 unseres Uno's als Puls-Weiten Modulationsausgang. Dieser Ausgang steuert dann ein IRF520 Treibermodul. Dieses Modul, das auch zum Beispiel zur Ansteuerung von Motoren genutzt wird, verwenden wir dazu, die Masseverbindung der eingebauten Leds des Displays zu steuern. (Der IRF520 schaltet gegen Masse durch)

Auf diese Weise können damit die Helligkeit unseres Displays in 255 Stufen steuern. Noch dazu auch noch bequem von unserem Bluetooth Terminal aus.

Doch nun zum eigentlichen Aufbau und zur Implementation der Erweiterung. Im ersten Schritt erweitern wir unsere Hardware wie in folgendem Schaltplan gezeigt, um das IRF520 MOS Driver Modul



In der Schaltung fällt als erstes auf, das die positiven Eingangs und Ausgangs Klemmen des IRF520 MOS Driver Moduls nicht belegt sind. Dies ist kein Schaltungsfehler, denn diese Klemmen / Anschlüsse des Moduls sind für das funktionieren unserer Schaltung nicht notwendig, da wir die Positive Betriebsspannung für die Hintergrundbeleuchtung des Displays über den Vorwiederstand direkt über die 5 Volt Schiene beziehen.

(Auch im Modul selbst sind die Klemmen Vin und V+ nur gebrückt und haben keine aktiven oder passiven Bauelemente zwischen den beiden Terminals!)

Lasst euch daher von der etwas vom Standard abweichenden Beschaltungsweise des Moduls nicht irritieren.

Als nächstes laden wir für diese Erweiterung auf unseren Arduino UNO folgenden angepassten Code hoch:

```
#include <SPI.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include <LiquidCrystal.h>
#include <avr/sleep.h>

#define MaxInputBufferSize 20 // maximal 255 Zeichen anpassen an vlcdr
#define EEpromSize 990

#define rLcdChr 20
#define LcdRows 4
#define interval 1000
#define BackgroundLight 5 // Port 5 Hintergrundbeleuchtung LED
#define DelayTOPWROFF 500

// EEprom SpeicherzellenAdressen für Konfiguration
#define EEFadeSeconds  993

#define EEPINA 996
#define EEPINC 997
#define EEPINDD 998


SoftwareSerial mySerial(7, 6); // RX, TX
LiquidCrystal lcd(8, 13, 12, 11, 10, 9);


//variables
byte DisplayBankContent = 0;

//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 SelectedMsg = 0;

//EEPROM
int eeaddress;                //EEPROM Adress Pointer
byte EEPromBanks = 0;         //Used for Calculating the EEPROM Banks
//SerMnueControl
byte MnuState = 0;            // Maximale Menuetiefe 255 icl Sub
byte Selectedbank =0;

//Real Time Clock
long previousMillis = 0;       // will store last time was measured
long previousMillisB = 0;      // will store last time was measured

//Display Management
boolean DisplayLock = false;
boolean Directprint = false;
boolean EchoMode = true;
byte DirectprintROW = 0;
byte DirectprintLine = 0;

boolean RefreshDisplay = false;
byte FRMCheck = 0; // Used fpr Writing Operations to eeprom so save Wirte cycles
```

```
// PWM Lichtsteuerung

byte Currentbrightness = 0;
byte Targetbrightness = 0;
byte FadeSeconds = 0; // Standard = 3



void setup()
{
  EEPromBanks = EEpromSize / ((rLcdChr) * LcdRows);
  lcd.begin(rLcdChr, LcdRows);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("    Bluetooth");
  lcd.setCursor(0, 1);
  lcd.print("     Display");
  mySerial.begin(9600);
  pinMode(BackgroundLight,OUTPUT); // Displaybeleuchtung / Display AN /AUS
  digitalWrite(BackgroundLight,LOW);
  // read Config
  FadeSeconds = 0;

 Currentbrightness = 0;
  Targetbrightness = 0;
  lcd.setCursor(0, 4);
  if (DisplayLock) {  lcd.print("  System gesperrt"); }
  // Further Setup Routines / initalizing
  lcd.setCursor(0, 0);
  Targetbrightness = 255;
  mySerial.flush();
}



//
##########################################################################################################
### //

void loop()

{
SerialcommandProcessor();
runrealTimeClock();
Displayprocessor();
//End Main loop
}

//
##########################################################################################################
### //


void TextHeader(byte rowm)
{
   mySerial.println("Text for Bank " + String( Selectedbank) + " ROW " + String (rowm) + ":");
  }

void SerialcommandProcessor()
{
int a;
Inptype = 0;
Inptype = SerInputHandler();
// 0 keine Rückgabe
// 1 Nummer
// 2 String
// 3 Data

if ((Inptype > 0) & (!Directprint))
   {
   MenueSelection = 0;
   if ((MnuState < 2) && (Inptype == 2)) {Sbuffer.toUpperCase(); } // For Easy Entering Commands
   if ((Sbuffer == "S") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 3;}
   // Erasing ALL EEprom Content
   if ((Sbuffer == "E") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 4;}
   if ((Sbuffer == "YES") && (MnuState == 1)&& (Inptype == 2))      { MenueSelection = 5;}
   if ((Sbuffer != "YES") && (MnuState == 1) && (Inptype == 2))     { MenueSelection = 6;}
```

```
//Edit  Selected Content
if ((Sbuffer == "W") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 7;}
if ((MnuState == 2) && (value  < EEPromBanks) && (Inptype == 1)) { MenueSelection = 8;}
if (MnuState == 3)                                               { MenueSelection = 9;}
if (MnuState == 4)                                               { MenueSelection = 10;}
//Display  Selected Content
if ((Sbuffer == "P") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 11;}
if ((MnuState == 5) && (Inptype == 1))                           { MenueSelection = 12;}
if ((Sbuffer == "R") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 13;}
if ((MnuState == 6) && (Inptype == 1))                           { MenueSelection = 14;}
if ((Sbuffer == "D") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 15;}
if ((Sbuffer == "Z") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 16;}
if ((Sbuffer == "B") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 17;}
if ((MnuState == 7) && (Inptype == 1))                           { MenueSelection = 18;}
if ((Sbuffer == "FADE") && (MnuState == 0) && (Inptype == 2))    { MenueSelection = 19;}
if (MnuState == 9)                                               { MenueSelection = 20;}
if (MnuState == 10)                                              { MenueSelection = 21;}
if (MnuState == 12)                                              { MenueSelection = 25;}
if (MnuState == 13)                                              { MenueSelection = 27;}
if (MnuState == 14)                                              { MenueSelection = 29;}

switch (MenueSelection)
 {
   case 3:
   {
     mySerial.println("Read EEEPROM Content:" );
     mySerial.flush();
     for (int a = 0; a < EEPromBanks; a++)
      {
        mySerial.println("EEPROM Memory Bank: " + String(a) );
        mySerial.flush();
        for (int b = 1; b <= LcdRows;b++)
          {
            mySerial.print("Row " + String(b) +": ");
            mySerial.flush();
            for (int c = 0; c <rLcdChr; c++)
               {
                 eeaddress = 0;
                 eeaddress = (a * (rLcdChr)* LcdRows) + ((rLcdChr) * b) + c;
                 value = EEPROM.read(eeaddress);
                 mySerial.print(char(value));
                 mySerial.flush();
               }
            mySerial.println(" ");
            mySerial.flush();
          }

      }
     Sbuffer = "";
     mySerial.println("No more EEPROM Banks available.");
     mySerial.flush();
     break;
   }
   case 4:
   {
     value = 0;
     mySerial.print("Erasing EEPROM ");
     mySerial.println("YES/NO:");
     mySerial.flush();
     MnuState = 1;
     Sbuffer = "";
     break;
   }
   case 5:
   {
    value = 0;
   mySerial.print("Erasing EEPROM ");
    mySerial.println("Stand by.");
    mySerial.flush();
    for (int a = 0; a < EEPromBanks; a++)
     {
       //Memory Bank a
       mySerial.println("Clear Bank: " + String(a));
       for (int b = 1; b <= LcdRows;b++)
        {
         for (int c = 0; c <rLcdChr; c++)
            {
```

```
              eeaddress = 0;
              eeaddress = (a * (rLcdChr)* LcdRows) + ((rLcdChr ) * b) + c;
              FRMCheck = EEPROM.read(eeaddress);
              if (FRMCheck > 0)
              {
              EEPROM.write(eeaddress,00); // Formatierung
              mySerial.print(".");
              value++;
              delay(30);
              mySerial.flush();
              }
             }
          }
        mySerial.println("");
        mySerial.flush();
      }
     mySerial.println("");
     mySerial.println("Finished. "+ String(value) + " Bytes cleared");
     mySerial.println("");
     mySerial.flush();
     Sbuffer = "";
     MnuState = 0;
     break;
    }
    case 6:
    {
     value = 0;
     Sbuffer = "";
     MnuState = 0;
     mySerial.println("OP abort.");
     mySerial.flush();
     break;
    }
    case 7:
    {
    mySerial.println("EEPPROM Bank Number (0-" + String(EEPromBanks-1) + "):");
    mySerial.flush();
    MnuState = 2;
    value = 0;
    Sbuffer = "";
    break;
    }
    case 8:
    {
    Selectedbank = value;
    TextHeader(1);

    MnuState = 3;
    Sbuffer = "";
    value = 0;
    break;
    }
    case 9:
    {
    WriteEEPROM(Selectedbank,1);
    TextHeader(2);
    value = 0;
    MnuState = 4;
    Sbuffer = "";
    break;
    }
    case 10:
    {
    WriteEEPROM(Selectedbank,2);
    value = 0;
    MnuState = 0;
    Sbuffer = "";
    TextHeader(3);
    mySerial.flush();
    value = 0;
    MnuState = 9;
    Sbuffer = "";
    break;
    }
    case 11:
    {
    value = 0;
```

```
mySerial.println("EEPPROM Bank Number (0-" + String(EEPromBanks-1) + "):");
 MnuState = 5;
 Sbuffer = "";
 mySerial.flush();
 break;
 }
 case 12:
 {
 SelectedMsg = value;
 DisplayBank(value);
 break;
 }
 case 13:
 {
 value = 0;
 mySerial.println("EEPPROM Bank Number (0-" + String(EEPromBanks-1) + "):");
 MnuState = 6;
 Sbuffer = "";
 mySerial.flush();
 break;
 }
 case 14:
 {
  a = value;
  if ( a < EEPromBanks)
  {
  mySerial.println("Memory Bank: " + String(a) );
  mySerial.flush();
  for (int b = 1; b <= LcdRows;b++)
    {
      mySerial.print("Row " + String(b) +": ");
      mySerial.flush();
      for (int c = 0; c <rLcdChr; c++)
        {
          eeaddress = 0;
          eeaddress = (a * (rLcdChr)* LcdRows) + ((rLcdChr) * b) + c;
          value = EEPROM.read(eeaddress);
          mySerial.print(char(value));
          mySerial.flush();
        }
      mySerial.println(" ");
      mySerial.flush();
    }
  } else
    {
     mySerial.println("Value out of Range.");
     }

  value = 0;
  Sbuffer = "";
  MnuState = 0;
  break;
 }
 case 15:
 {
 // Direct pPrint to Display
 Directprint = true;
 mySerial.println ("Directprint ON.");
 if (Directprint)
 {
 DirectprintROW = 0;
 DirectprintLine = 0;
 lcd.clear();
 lcd.cursor();
 lcd.blink();
 }
 value = 0;
 Sbuffer = "";
 MnuState = 0;
 break;
 }
 case 16:
 {
 value = 0;
 Sbuffer = "";
 MnuState = 0;
 break;
```

```
}
case 17:
{
mySerial.println("Display Brightness: (max 255)");
MnuState = 7;
value = 0;
Sbuffer = "";
break;
}
case 18:
{
if ((value < 256))
  {
    Targetbrightness = value;
    mySerial.println("Brightness: " + String (Targetbrightness) + " Set");
  } else
  {
    mySerial.println("Value out of Range.");
     }
MnuState = 0;
value = 0;
Sbuffer = "";
break;
}
case 19:
{
mySerial.println("Fade Delay: (max 255 Sec)");
MnuState = 12;
value = 0;
Sbuffer = "";
break;
}
case 20:
{
WriteEEPROM(Selectedbank,3);
value = 0;
MnuState = 0;
Sbuffer = "";
TextHeader(4);
mySerial.flush();
value = 0;
MnuState = 10;
Sbuffer = "";
break;
}
case 21:
{
WriteEEPROM(Selectedbank,4);
value = 0;
MnuState = 0;
Sbuffer = "";
break;
}
case 25:
{
 if ((value > 0) & (value < 251))
{
FadeSeconds = value;
EEPROM.write(EEFadeSeconds, FadeSeconds);
mySerial.println("Value " + String (value) + " set.");
} else
{
value = 0;
Sbuffer = "";
mySerial.println("Value out of Range.");
}
value = 0;
MnuState = 0;
Sbuffer = "";
break;
}

default:
{
  mySerial.println("-------Smart Bluetooth Display 1.1------");
  mySerial.println("S - Read ALL EEPROM Banks");
  mySerial.println("E - Erase ALL EEPROM Banks");
```

```
        mySerial.println("W - Write sel. EEPROM Bank");
        mySerial.println("R - Read sel. EEPROM Bank");
        mySerial.println("P - Print EEPROM Bank on Display");
        mySerial.println("----------------------------------------");
        mySerial.println("D - Direct Print");
        mySerial.println("B - Display Brightness Current Value: " + String (Currentbrightness));
        mySerial.println("----------------------------------------");
        mySerial.println("Type Cmd and press Enter");
        mySerial.flush();
        MnuState = 0;
        value = 0;
        Sbuffer = "";
       }
    }
   } // Eingabe erkannt
}



void WriteEEPROM(byte FBank,byte FRow)
{
byte Writecounter;
Writecounter = 0;
 mySerial.print("Saving ");
       for (int c = 0; c <rLcdChr; c++)
         {
          eeaddress = 0;
          eeaddress = (FBank * (rLcdChr)* LcdRows) + ((rLcdChr) * FRow) + c;

          value = EEPROM.read(eeaddress);
          if (Sbuffer[c] != value)
          {
          EEPROM.write(eeaddress,Sbuffer[c]);
          mySerial.print(".");
          Writecounter++;
          }
         }
       mySerial.println(" " + String (Writecounter) + " Bytes written.");

}

void ClearCBuffer ()

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

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
}

// Eingabebuffer
boolean CheckforserialEvent()
{
  while (mySerial.available()) {
     // get the new byte:
     TBuffer = mySerial.read();
     if (TBuffer > 9 && TBuffer < 14)
       {
        Cbuffer[Ccount] = 0;
        TBuffer =0;
        if (EchoMode)
        {
         mySerial.print(char(13));
         mySerial.flush();
        }
        if (Directprint)
        {
        mySerial.println("");
        DirectprintLine = 0;
        DirectprintROW = DirectprintROW + 1;
        if ( DirectprintROW > 3)
        {
                Directprint = false;
                lcd.noCursor();
                lcd.noBlink();
                Sbuffer = "";
                value = 0;

          } else
        {
        lcd.cursor();
        lcd.blink();
        lcd.setCursor(0,DirectprintROW);
        }
        }

        EnterInput = true;
        return true;
     } else if (TBuffer > 47 && TBuffer <58 )
      {
       if ( Ccount < MaxInputBufferSize)
        {
          Cbuffer[Ccount] = TBuffer;
          Ccount++;
          if ((Directprint))
           {
             lcd.print(char(TBuffer));
             DirectprintLine = DirectprintLine + 1;
             if ( Ccount > MaxInputBufferSize -1)
             {
               lcd.noCursor();
               lcd.noBlink();
               } else {
               lcd.cursor();
               lcd.blink();
               }

           }
          if (EchoMode) {
          mySerial.print(char(TBuffer));
          mySerial.flush();
          }
        } else {mySerial.print("#"); }

      //Number Input detected
      NumberInput = true;
      }
      else if (TBuffer > 64 && TBuffer < 123 )
```

```
    {
     if ( Ccount < MaxInputBufferSize)
       {
         Cbuffer[Ccount] = TBuffer;
         Ccount++;
         if ((Directprint))
          {
            lcd.print(char(TBuffer));
            DirectprintLine = DirectprintLine + 1;
            if ( Ccount > MaxInputBufferSize -1)
            {
              lcd.noCursor();
              lcd.noBlink();
              } else {
              lcd.cursor();
              lcd.blink();
              }
          }
         if (EchoMode) {
         mySerial.print(char(TBuffer));
         mySerial.flush();
         }
       } else {mySerial.print("#"); }
   StrInput = true;
  }
 else if ( (TBuffer == 127 )  |  (TBuffer == 8 ) )
     {


            if ( DirectprintLine > 0 )
            {
            DirectprintLine = DirectprintLine - 1;
            lcd.setCursor(DirectprintLine, DirectprintROW);
            lcd.print(" ");
            lcd.setCursor(DirectprintLine, DirectprintROW);
            }


            if (( DirectprintLine == 0 ) & ( DirectprintROW > 0 ))
            {
            DirectprintROW = DirectprintROW - 1;
            DirectprintLine = rLcdChr -1;
            lcd.setCursor(DirectprintLine, DirectprintROW);

            }

     if ( Ccount > 0)
      {
        Ccount--;
        Cbuffer[Ccount] = 0;
        if ((Directprint))
         {

            if ( Ccount > MaxInputBufferSize -1)
            {
              lcd.noCursor();
              lcd.noBlink();
              } else {
              lcd.cursor();
              lcd.blink();
              }
         }


        if (EchoMode) {
        mySerial.print("-");
        mySerial.flush();
        }
      }
  }
 else
   {

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

```
            if ((Directprint))
            {
              DirectprintLine = DirectprintLine + 1;
              if (TBuffer < 128) {lcd.print(char(TBuffer)); } else  {lcd.print(String(TBuffer)); }
              if ( Ccount > MaxInputBufferSize -1)
              {
               lcd.noCursor();
               lcd.noBlink();
               } else {
               lcd.cursor();
               lcd.blink();
               }
            }
           if (EchoMode) {
           mySerial.print(char(TBuffer));
           mySerial.flush();
           }
         } else {mySerial.print("#"); }

      //Data Input detected
      DataInput = true;
      }
     return false;
     }
  return false;
  }

void Displayprocessor()  // Bei Blauem Display wird auf Scrollfunktion verzichtet, da das nur "schmiert"
{
if (RefreshDisplay)
  {
    lcd.clear();
    RefreshDisplay = false;
       for (int b = 1; b <= LcdRows;b++)
              {
               lcd.setCursor(0, b -1);
               for (int c = 0; c <rLcdChr; c++)
                  {
                   eeaddress = 0;
                   eeaddress = (DisplayBankContent * (rLcdChr)* LcdRows) + ((rLcdChr) * b) + c;
                   value = 0;
                   value = EEPROM.read(eeaddress);
                   if (value > 31) // Sonderzeichen nicht anzeigen
                   {
                    mySerial.print(char(value));
                    lcd.print(char(value));
                    } else
                  { lcd.print(char(32)); }
                  }
                 mySerial.println();
              }
   }
}

void runrealTimeClock()    //TIMEBASE
{
  // Real Time Clock & Countdown
 // long previousMillis = 0;       // will store last time was measured
 // byte SecDivider = 0;
unsigned long currentMillis = millis();

int StepValue = 0;
// PWM Display Steuerung
 StepValue = 4 * FadeSeconds;
  if(currentMillis - previousMillis > StepValue)
  {
     previousMillis = currentMillis;

  if (Currentbrightness < Targetbrightness
  )
  {
  Currentbrightness = Currentbrightness + 1;
  analogWrite (BackgroundLight,Currentbrightness);
  } else if (Currentbrightness > Targetbrightness)
   {
  Currentbrightness = Currentbrightness - 1;
  analogWrite (BackgroundLight,Currentbrightness);
```

```
   }
   }
  if(currentMillis - previousMillisB > 1000)
  {
  // sekundentakt
    previousMillisB = currentMillis;
   }
}

void DisplayBank ( byte cobank)
{
     if (cobank  < EEPromBanks )
     {
     RefreshDisplay = true;   // Initalize Display Output
     DisplayBankContent = cobank;
     mySerial.println("Bank " + String(cobank) + " is displayed on LCD");
     MnuState = 0;
     Sbuffer = "";
     value =0;
     mySerial.flush();
     } else
          {
            mySerial.println("Bank not available.");
            value = 0;
            MnuState = 0;
            Sbuffer = "";
            mySerial.flush();
          }
}
```

Fertig ! Mehr ist nicht mehr notwendig.

Nach dieser kleinen Erweiterung bzw. Anpassung der Hardware steht uns nun der weitere Befehl „B“ für Brightness in dem Bluetooth Menü zur Verfügung.

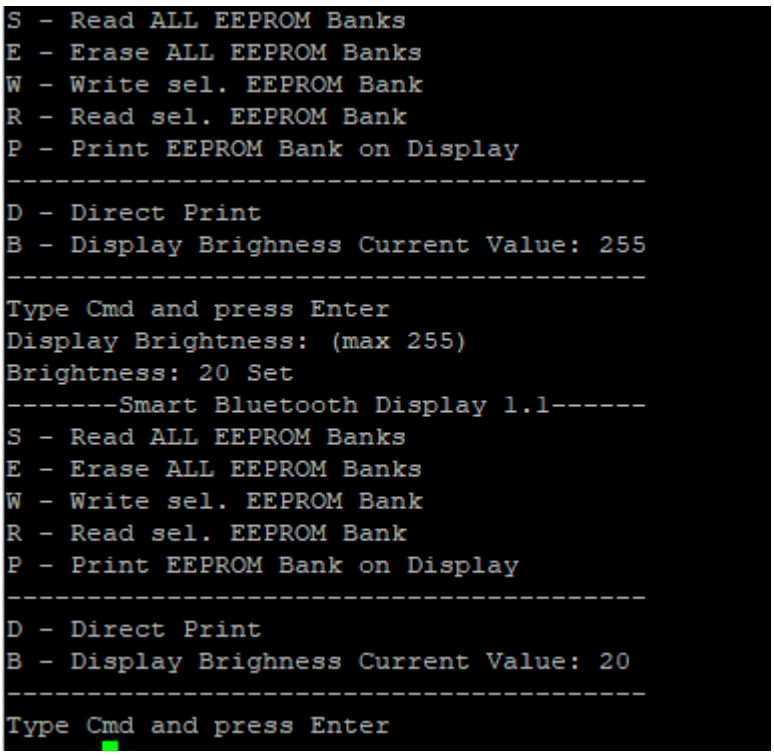
- B – Display Brightness Current Value: 255

```
-------Smart Bluetooth Display 1.1------
S - Read ALL EEPROM Banks
E - Erase ALL EEPROM Banks
W - Write sel. EEPROM Bank
R - Read sel. EEPROM Bank
P - Print EEPROM Bank on Display
---------------------------------------
D - Direct Print
B - Display Brighness Current Value: 255
---------------------------------------
Type Cmd and press Enter
Display Brightness: (max 255)
Brightness: 20 Set
```

Die dahinter angegebene Zahl beschreibt die aktuell eingestellte Displayhelligkeit in einem Wertebereich von 0 (Hintergrundbeleuchtung Aus) bis maximal 255 (maximale Helligkeit)

So sehen wir immer auf einem Blick, welche Helligkeit gerade auf unserem Display eingestellt ist.

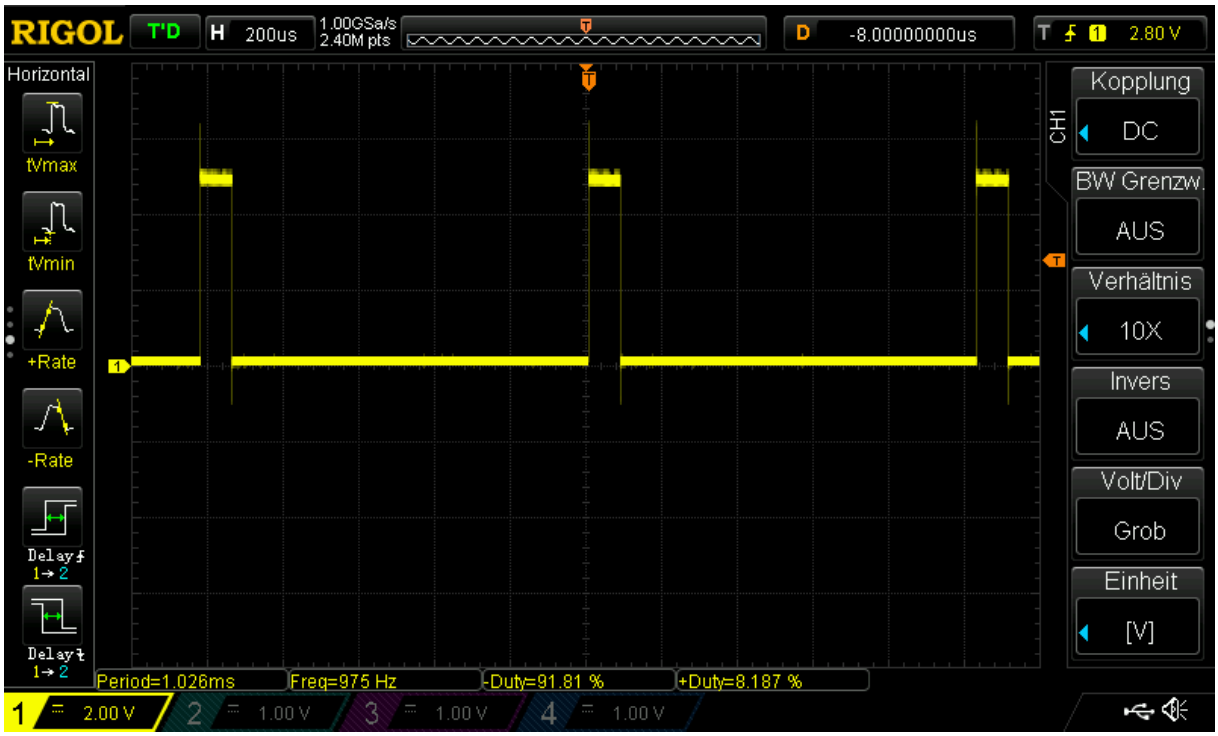
Möchten wir nun die Helligkeit der Hintergrundbeleuchtung ändern, so wählen wir den Punkt „B“ (Brightness) aus und drücken Enter. Wir werden dann kurz und knapp gefragt: Display Brightness: (max 255) und geben dahinter unseren gewünschten Wert ein.



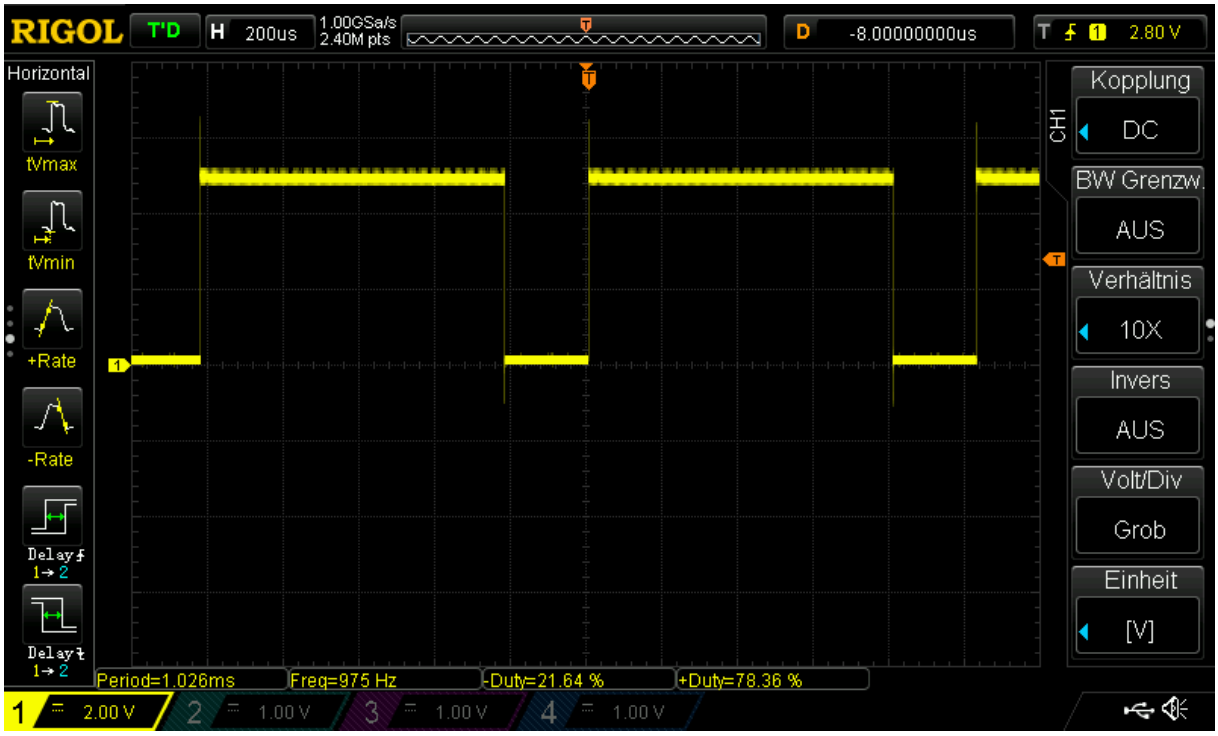
(Wir geben In diesem Beispiel 20 als gewünschten Helligkeitswert ein)

Dies wird mit der Ausgabe: Brightness: 20 Set quittiert.Wir sehen nun, wie das Display sich auf den gewünschten Wert verdunkelt.

Für alle Interessierten, die das PWM Signal, das unser Code zur Steuerung der Helligkeit generiert, sehen möchten habe ich von meinem Oszilloskop einmal bei dem Wert 20 und bei dem Wert 200 Screenshots angefertigt:



Wert:20



Wert: 200

Wie zu erkennen ist, unterscheidet sich das Puls-Pausen Verhältnis des Signales, während die Frequenz gleichbleibt. Unser Auge mittelt dieses Verhältnis wieder zu einer gleichmäßigen Helligkeit.

Ich wünsche viel Spaß beim Nachbauen und wie immer bis zum nächsten Mal.