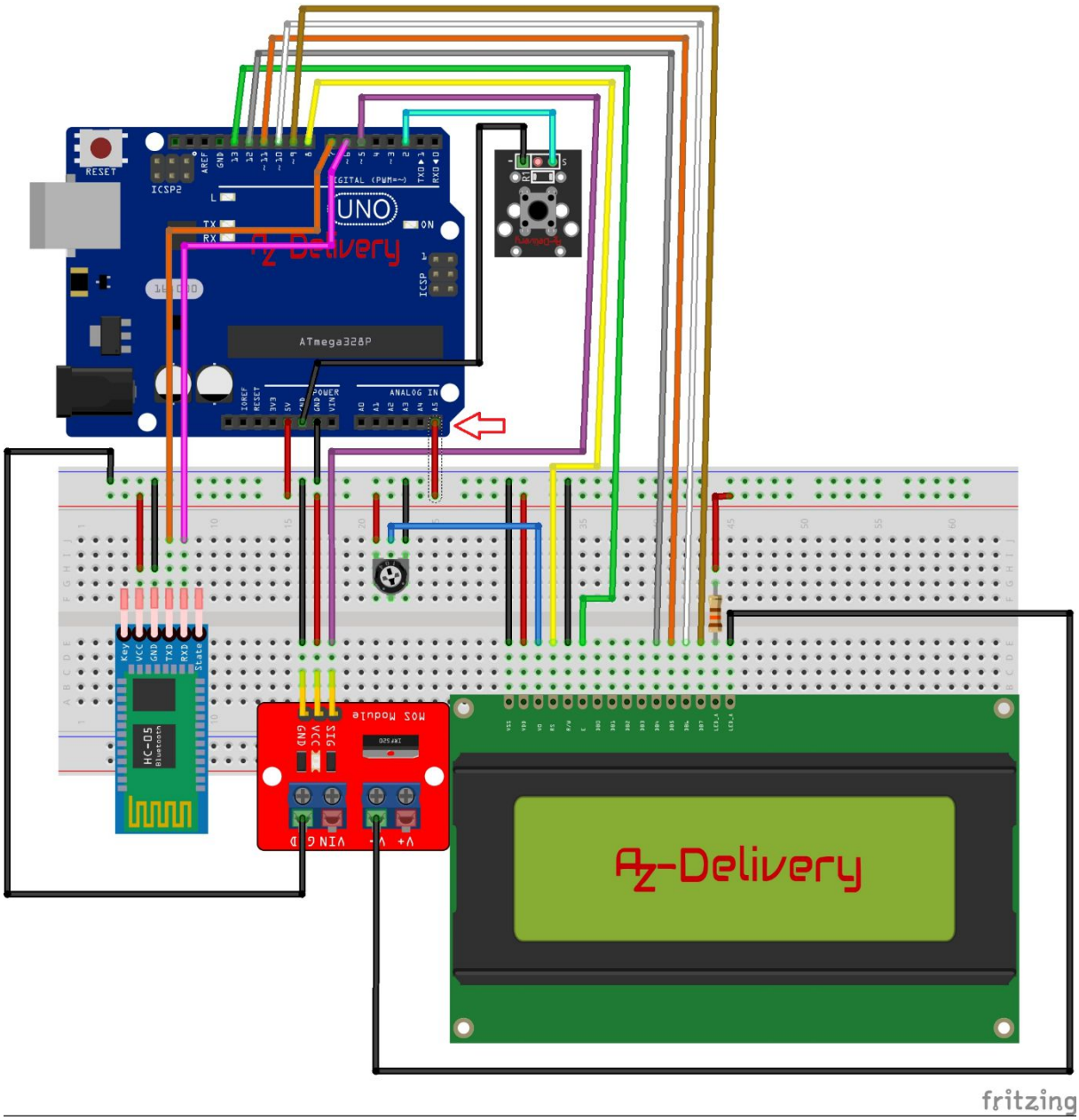
Willkommen zum sechsten und letzten Teil unserer Bluetooth Display Reihe. Auch in diesem Teil habe ich mir wieder etwas Nettes als Ergänzung zu den bisherigen Funktionen ausgedacht. Im heutigen Teil werden wir uns dem Energie und Batterie Management der Einheit widmen. Dies als Ergänzung zum letzten Teil, bei dem wir unser Display als Werbetafel auf dem Flohmarkt eingesetzt haben. Auf dem Flohmarkt gibt es selten Stecker, sodass wir unser Display mit einer Powerbank betreiben müssen. Was aber wenn uns die Energie ausgeht, oder wir wissen wollen, ob die Versorgungspannung noch 5 Volt beträgt? Dies geht am besten, indem wir einen der Analog Eingänge zur Messung der Batteriespannung nutzen und direkt in Volt die Betriebsspannung auf unserem Display anzeigen.

Außerdem wäre eine Energiesparfunktion sinnvoll, die wir mit den bisher eigebauten Steuerelementen (Taster) bedienen können. Des Weiteren bekommt unser Display nun abschließend noch eine Debug-Funktion spendiert, die uns hilft, Fehler auf technischer Ebene zu finden und eine Erleichterung bei Erweiterungen des Codes mit sich bringt.

Damit aber unser Display erst einmal die Spannung der Versorgung überhaut erfassen kann, ist es notwendig eine weitere kleine Hardwareerweiterung vorzunehmen. Diese Erweiterung besteht aus der Verbindung des Analog Port 5 mit der Betriebsspannung VDD. Nachfolgend ist die aus einer Drahtbrücke bestehende Verbindung im Schaltbild abgebildet und durch einen roten Pfeil markiert



Auch in diesem Teil nehmen wir eine Erweiterung bzw. Anpassung des Firmwarequellcodes vor und laden daher folgenden Code auf den Arduino 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 VoltageSensingPin 5 //
#define SwitchPin 2 // Port 12 Taster Nachrichtsauswahl
```

```
#define DelayTOPWROFF 500
#define MinOPVoltage 3.4

// EEprom SpeicherzellenAdressen für Konfiguration
#define EEFadeSeconds  993
#define EEAdvertsecdelay 994
#define EEAdvertMsg 995
#define EEPINA 996
#define EEPINC 997
#define EEPINDD 998

SoftwareSerial mySerial(7, 6); // RX, TX

//LiquidCrystal(rs, enable, d4, d5, d6, d7)
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;

//Druckknopfsteuerung
boolean Switchstate = true;
boolean SwitchstateBuffer = true;
byte SelectedMsg = 0;

//Give Debug Informations over serial Interface
boolean DebugMode = false;
boolean EchoMode = true;

//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;
byte DirectprintROW = 0;
byte DirectprintLine = 0;

boolean RefreshDisplay = false;
byte FRMCheck = 0; // Used fpr Writing Operations to eeprom so save Wirte cycles
// BatterieMonitoring
float Voltage;
boolean PowersaveMode = false;

// PWM Lichtsteuerung

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

// Auto Display Z.B für Werbungszwecke

boolean Advertising = false;
byte AdvertMsg = 1; // Minimum 1
byte Advertsecdelay = 0; // Standard = 6
byte Advertseccounter = 0;
```

```
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(SwitchPin,INPUT_PULLUP); // Taster Auswahl Text aus EEprom
  pinMode(BackgroundLight,OUTPUT); // Displaybeleuchtung / Display AN /AUS
  digitalWrite(BackgroundLight,LOW);
  // read Config
  FadeSeconds = EEPROM.read(EEFadeSeconds);
  Advertsecdelay =EEPROM.read(EEAdvertsecdelay);
  AdvertMsg =EEPROM.read(EEAdvertMsg);
  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();
SwitchProcessor();
//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))

   {
   if (PowersaveMode) {
   TogglePowerSave(false);
   }
   MenueSelection = 0;
   if (Advertising)
   {
     lcd.clear();
     Advertising = false;
     mySerial.print(F("Advertising "));
     mySerial.println(F(" OFF."));
     }
   if ((MnuState < 2) && (Inptype == 2)) {Sbuffer.toUpperCase(); } // For Easy Entering Commands
```

```
  if ((Sbuffer == "DEBUG") && (MnuState == 0) && (Inptype == 2))   { MenueSelection = 1;}
  if ((Sbuffer == "ECHO")&& (MnuState == 0) && (Inptype == 2))       { MenueSelection = 2;}
  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 ((Sbuffer == "A") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 22;}
  if ((Sbuffer == "ADVERT") && (MnuState == 0) && (Inptype == 2))  { MenueSelection = 23;}
  if (MnuState == 11)                                              { MenueSelection = 24;}
  if (MnuState == 12)                                              { MenueSelection = 25;}
  if ((Sbuffer == "ADSEC") && (MnuState == 0) && (Inptype == 2))   { MenueSelection = 26;}
  if (MnuState == 13)                                              { MenueSelection = 27;}
  if ((Sbuffer == "ADMSG") && (MnuState == 0) && (Inptype == 2))   { MenueSelection = 28;}
  if (MnuState == 14)                                              { MenueSelection = 29;}
  switch (MenueSelection)
  {
     case 1:
     {
       mySerial.print("Debug ");
       DebugMode = !DebugMode;
       if (DebugMode) {
       mySerial.println(F(" ON."));
       EchoMode = false;
        } else
        {
          mySerial.println(F(" OFF."));
          EchoMode = true;
        }
       mySerial.println("");
       mySerial.flush();
       value = 0;
       MnuState = 0;
       Sbuffer = "";
       break;
     }
     case 2:
     {
        mySerial.print(F("Echo "));
        EchoMode = !EchoMode;
        if (EchoMode) {
          mySerial.println(F(" ON."));
          DebugMode = false;
          } else
          {
          mySerial.println(F(" OFF."));
          }
        mySerial.println(F(""));
        mySerial.flush();
        value = 0;
        MnuState = 0;
        Sbuffer = "";
        break;
     }
     case 3:
     {
       mySerial.println(F("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(F(" "));
              mySerial.flush();
            }

       }
      Sbuffer = "";
      mySerial.println(F("No more EEPROM Banks available."));
      mySerial.flush();
      break;
    }
    case 4:
    {
      value = 0;
      mySerial.print("Erasing EEPROM ");
      mySerial.println(F("YES/NO:"));
      mySerial.flush();
      MnuState = 1;
      Sbuffer = "";
      break;
    }
    case 5:
    {
     value = 0;
    mySerial.print(F("Erasing EEPROM "));
     mySerial.println(F("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(F("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(F("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(F("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(F("Value out of Range."));
     }
MnuState = 0;
value = 0;
Sbuffer = "";
break;
}
case 19:
{
mySerial.println(F("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 22:
{
MnuState = 0;
Sbuffer = "";
mySerial.println(F("Enter NEW BT Pin"));
mySerial.flush();
MnuState = 11;
break;
}
case 23:
{
Advertising  = !Advertising;
mySerial.print(F("Advertising "));
if (Advertising)
 {
 mySerial.println(F(" ON."));
 } else  { mySerial.println(F(" OFF.")); }
mySerial.flush();

value = 0;
MnuState = 0;
Sbuffer = "";
break;
}

case 24:
{
 if ((value > 1000) & (value < 10000))
{
mySerial.print("NEW PIN "+ Sbuffer);
delay(1000);
mySerial.flush();
delay(1000);
mySerial.println("AT+PIN"+ Sbuffer);
delay(3000);
mySerial.flush();
} else
{
value = 0;
Sbuffer = "";
mySerial.println(F("Value out of Range."));
}
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(F("Value out of Range."));
  }
  value = 0;
  MnuState = 0;
  Sbuffer = "";
  break;
  }
  case 26:
  {
 mySerial.println(F("ADverDelay: (max 255 Sec)"));
  MnuState = 13;
  value = 0;
  Sbuffer = "";
  break;
  }
  case 27:
  {
   if ((value > 0) & (value < 256))
  {

  Advertsecdelay = value;
  EEPROM.write(EEAdvertsecdelay,Advertsecdelay);
mySerial.println("Value " + String (value) + " set.");
  } else
  {
  value = 0;
  Sbuffer = "";
  mySerial.println(F("Value out of Range."));
  }
  value = 0;
  MnuState = 0;
  Sbuffer = "";
  break;
  }
  case 28:
  {
 mySerial.println("ADMsgs: (max " + String (EEPromBanks) + ")");
 MnuState = 14;
  value = 0;
  Sbuffer = "";
  break;
  }
  case 29:
  {
   if ((value > 0) & (value < EEPromBanks +1))
  {

  AdvertMsg = value;
  EEPROM.write(EEAdvertMsg,AdvertMsg);
mySerial.println("Value " + String (value) + " set.");
  } else
  {
  value = 0;
  Sbuffer = "";
  mySerial.println(F("Value out of Range."));
  }
  value = 0;
  MnuState = 0;
  Sbuffer = "";
  break;
  }
  default:
  {
    if (DisplayLock)
      {
       lcd.clear();
       DisplayLock = false;
      }
    mySerial.println(F("-------Smart Bluetooth Display 1.1------"));
```

```
        mySerial.println(F("S - Read ALL EEPROM Banks"));
        mySerial.println(F("E - Erase ALL EEPROM Banks"));
        mySerial.println(F("W - Write sel. EEPROM Bank"));
        mySerial.println(F("R - Read sel. EEPROM Bank"));
        mySerial.println(F("P - Print EEPROM Bank on Display"));
        mySerial.println(F("----------------------------------------"));
        mySerial.println(F("D - Direct Print"));
        mySerial.println("B - Display Brighness Current Value: " + String (Currentbrightness));
        mySerial.println(F("----------------------------------------"));
        mySerial.println(F("A - Bluetooth Access PIN"));
        mySerial.println(F("Other: ADVERT,ADSEC,ADMSG,FADE,ECHO"));
        mySerial.println(F("----------------------------------------"));
        mySerial.println(F("Type Cmd and press Enter"));
        mySerial.flush();
        if (DebugMode)
         {
          mySerial.println("Debug: CmdProc End Typ: " + String(Inptype) + " Inhalt: " +Sbuffer);
          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;
         if (DebugMode)
          {
           mySerial.println("Debug: Number: "+String(value));
           mySerial.flush();
          }
         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;
       if (DebugMode)
          {
            mySerial.println("Debug: String: "+Sbuffer);
            mySerial.flush();
          }
      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;
      if (DebugMode)
        {
          mySerial.println("Debug: Data: "+Sbuffer);
          mySerial.flush();
        }
      Ccount = 0;
      ClearCBuffer;
      result = 3;                                   //Number Returncode
      }
      if ((EnterInput) and not (StrInput) and not (NumberInput) and not (DataInput))
      {
      Sbuffer = "";
      value = 0;
      if (DebugMode)
        {
          mySerial.println(F("Debug: Only Enter pressed"));
          mySerial.flush();
        }
      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);
         }
         }
        if (DebugMode)
         {
          mySerial.println("Debug: Enter received. Length: " + String(Ccount));
          mySerial.flush();
```

```
        }
       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("#"); }
    if (DebugMode) {
    mySerial.println(F("Debug: Number over Serial received "));
    mySerial.flush();
    }
    //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("#"); }

    if (DebugMode) { mySerial.println(F("Debug: Char over Serial received "));
    mySerial.flush(); }
    //Character Char Input detected
    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("#"); }
       if (DebugMode)
        {
          mySerial.println(F("Debug: Data over Serial received "));
          mySerial.flush();
        }
      //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);
                if (!Advertising) {mySerial.print("Row " + String(b) +": "); }
```

```
                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
                    {
                     if (!Advertising) { mySerial.print(char(value)); } else { delay(100);}
                     lcd.print(char(value));
                     } else
                    { lcd.print(char(32)); }
                   }
                 if (!Advertising) { 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;
    BattMonitoring();
// Advertising
    if (Advertising)
    {
     if (Advertseccounter >  Advertsecdelay)
      {
      Advertseccounter = 0;
      DisplayBankContent = DisplayBankContent + 1;
      if (DisplayBankContent > AdvertMsg - 1) { DisplayBankContent = 0; }
      RefreshDisplay = true;
      } else { Advertseccounter = Advertseccounter +1; }
      }
    }
}

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(F("Bank not available."));
               value = 0;
```

```
              MnuState = 0;
              Sbuffer = "";
              mySerial.flush();
            }
 }

void TogglePowerSave(boolean state)
{
mySerial.print("Powersave Mode ");
 if (!state)
   {
   PowersaveMode = false;
   Targetbrightness = 255;
   mySerial.println(F("OFF"));
   } else
   {
   PowersaveMode = true;
   Targetbrightness = 0;
   mySerial.println(F("ON"));
   }
}

void SwitchProcessor()
{
Switchstate = digitalRead(SwitchPin);
if ((!Switchstate) && (SwitchstateBuffer) && (not DisplayLock))// Abfrage Schalter
  {
   SwitchstateBuffer = false;
   Advertising = false;
   Directprint = false;
   lcd.noCursor();
   lcd.noBlink();
   if (PowersaveMode)
   {
   TogglePowerSave(false);
   } else
   {
   SelectedMsg = SelectedMsg + 1;
   if (SelectedMsg >  EEPromBanks - 1 )
     {
     SelectedMsg = 0;
     }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Bank: " + String(SelectedMsg) + " selected");
    lcd.setCursor(0,2);

    lcd.print("System VDD: " + String(Voltage) + " V");
    delay(10);
    value = DelayTOPWROFF;

    while (digitalRead(SwitchPin) == 0)
    {
     delay(1);
     if (value > 0) {value = value - 1;};
     if (value == 0 &&  not PowersaveMode)
     {
     TogglePowerSave(true);
    // if (SelectedMsg = 0 ) { SelectedMsg =  EEPromBanks;} else { SelectedMsg = SelectedMsg - 1;};
     lcd.setCursor(0,3);
     lcd.print("Power off OK    ");
     }

     lcd.setCursor(0,3);
     if (value > 0) {lcd.print("Power off: " + String(value /100)+ " sec ");};
    }
    DisplayBank(SelectedMsg);
  }

  }

if (Switchstate)
    {
     SwitchstateBuffer = true;
   //  delay(10);
    }
}
```

```
void Powerdown ()  // Toggle Powerdown, if critical Voltage is reached.
{
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Battery LOW Warning");
 lcd.setCursor(0,2);
 lcd.print("Charge Battery.");
 Directprint = false;
 Advertising = false;
 delay(5000);
 TogglePowerSave(true);
}

void BattMonitoring()

{
int BatteryState;

  BatteryState = analogRead(VoltageSensingPin);    // read the input pin
 // Mittelung der Messwerte noch hinzufügen

  Voltage = BatteryState * (0.00489);
 if (Voltage <= MinOPVoltage)
{
Powerdown();
}

}
```

Auch wenn sich auf den ersten Blick nicht viel getan hat im Hauptmenü, stecken die Erweiterungen „unter der Motorhaube“. Die erste große Erweiterung fällt auf, wenn wir nun den Taster zum Wechseln der Anzeigetexte drücken und gedrückt halten

Es wird jetzt nicht nur, wie vorher, die aktuell ausgewählte Text Bank angezeigt, sondern darüber hinaus jetzt die aktuelle Betriebsspannung. (im Bild untern 4,97 Volt) sowie ein Counter, der beginnend mit 5 herunterzählt und bei Erreichen von 0 das System herunterfährt. Ebenso wird das System bei unterschreiben der im Quelltext angegeben Minimalbetriebsspannung heruntergefahren. Aktuell wird das Display zwar geleert, und die Hintergrundbeleuchtung abgeschaltet, es bleibt aber noch grundsätzlich mit Spannung versorgt. Wer möchte kann das Display per Feldeffekt Transistor auch noch spannungsfrei schalten. Ich habe aus Gründen der Einfachheit jedoch hierauf verzichtet.

Im Menü selbst ginbt es jetzt den „versteckten“ Befehl „debug“ der den Debug Modus durch einfache Eingabe des Befehls ein- und ausschaltet. Der nachfolgende Screenshot aus der Handy APP „BlueTerm“ die ich für dieses Projekt sehr empfehlen kann, sieht man die Zusatzausgaben (Markiert mit dem Prefix „Debug:“) die ausgegeben werden, wenn der Modus eingeschaltet ist. Im Screenshot sieht man die Debugausgabe aller über Serielle Schnittstelle empfangenen Zeichen. (Hier debug)

Nach Empfang des Befehls debug wird der Debug Modus ausgeschaltet und mit „Debug OFF“ quittiert.

```
-------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
----------------------------------------
A - Bluetooth Access PIN
Other: ADVERT,ADSEC,ADMSG,FADE,ECHO
----------------------------------------
Type Cmd and press Enter
Debug: CmdProc End Typ: 2 Inhalt: TES
Debug: Data over Serial received
Debug: Enter received. Length: 1
Debug: Data:
-------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
----------------------------------------
A - Bluetooth Access PIN
Other: ADVERT,ADSEC,ADMSG,FADE,ECHO
----------------------------------------
Type Cmd and press Enter
Debug: CmdProc End Typ: 3 Inhalt:
Debug: Char over Serial received
Debug: Char over Serial received
Debug: Char over Serial received
Debug: Char over Serial received
Debug: Char over Serial received
Debug: Enter received. Length: 5
Debug: String: debug
Debug  OFF.
```

Ich wünsche viel Spaß beim Nachbauen. Bis bald beim nächsten Blog und beim nächsten Projekt. Ich hoffe Ihr hattet so ciel Spaß beim Nachbauen wie ich beim entwickeln.