Willkommen zu dem fünften und vorletzten Teil der Bluetooth Display Reihe. In dem heutigen Teil habe ich mir etwas besonderes einfallen lassen. Die Idee kam mir bei einem Flohmarkt, bei den ich ein paar alte Sachen verkaufen wollte. Ich überlegte mir, dass es doch eine gute Idee währe das Display nicht zuhause verwenden zu können, sondern dass das Display auch als „Eye Catcher“ bei einem Flohmarkt zu verwenden. Schließlich hat nicht jeder so etwas, und im regulären Handel habe ich ein ähnliches Display auch nicht gefunden. Doch was soll ich auf dem Display bei einem Flohmarkt anzeigen? Am besten einen Verkaufsticker! Wie währe es mit Botschaften wie: Jede CD nur xx Euro, gefolgt von einer weiteren Nachricht wie: Jede DVD xx Euro. Das Ganze sieht natürlich am besten als Ticker Laufschrift aus, daher bauen sich langsam die einzelnen Buchstaben von linke nach rechts auf, bis die ganze Nachricht lesbar ist. Für jeden Flohmarkt ein totaler Hingucker! Um diese Funktion zu realisieren erweitern wir nun unseren Code um die „Advertising“ Funktion.

Wir laden daher zuerst den um die Funktion „Advertising“ erweiterten Code auf unsere Hardwareplattform 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 SwitchPin 2 // Port 12 Taster Nachrichtsauswahl

// 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 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();

//Powermgmt();
//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 (Advertising)
   {
     lcd.clear();
     Advertising = false;
     mySerial.print("Advertising ");
     mySerial.println(" 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 == "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(" ON.");
       EchoMode = false;
        } else
        {
          mySerial.println(" OFF.");
          EchoMode = true;
        }
       mySerial.println("");
       mySerial.flush();
       value = 0;
       MnuState = 0;
       Sbuffer = "";
       break;
     }
     case 2:
     {
        mySerial.print("Echo ");
        EchoMode = !EchoMode;
        if (EchoMode) {
          mySerial.println(" ON.");
          DebugMode = false;
          } else
          {
          mySerial.println(" OFF.");
          }
        mySerial.println("");
        mySerial.flush();
        value = 0;
        MnuState = 0;
        Sbuffer = "";
        break;
     }
     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 23:
{
Advertising  = !Advertising;
mySerial.print("Advertising ");
if (Advertising)
 {
 mySerial.println(" ON.");
 } else  { mySerial.println(" OFF."); }
mySerial.flush();

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;
}
case 26:
{
mySerial.println("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("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("Value out of Range.");
       }
       value = 0;
       MnuState = 0;
       Sbuffer = "";
       break;
       }

       default:
       {
         if (DisplayLock)
           {
            lcd.clear();
            DisplayLock = false;
           }
         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 Brighness Current Value: " + String (Currentbrightness));
         mySerial.println("----------------------------------------");
         mySerial.println("Other: ADVERT,ADSEC,ADMSG,ECHO");
         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)
```

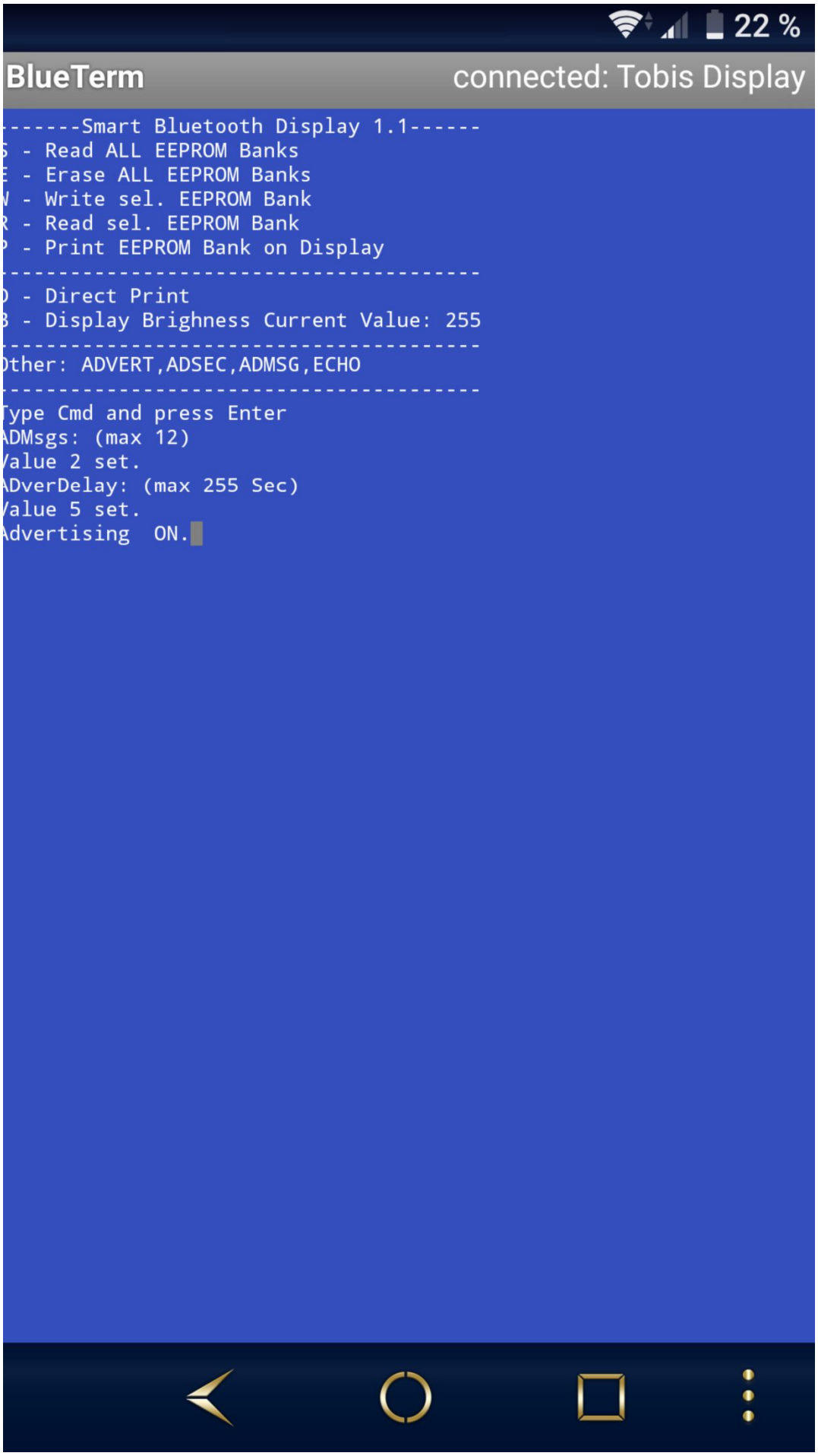
```
      {
      mySerial.println("");
      DirectprintLine = 0;
      DirectprintROW = DirectprintROW + 1;
      if ( DirectprintROW > 3)
      {
            Directprint = false;
            lcd.noCursor();
            lcd.noBlink();
            Sbuffer = "";
            value = 0;

       } 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("#"); }
    //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("#"); }
      //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;
// 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("Bank not available.");
              value = 0;
              MnuState = 0;
              Sbuffer = "";
              mySerial.flush();
            }
 }

void SwitchProcessor()
{
Switchstate = digitalRead(SwitchPin);
if ((!Switchstate) && (SwitchstateBuffer) && (not DisplayLock))// Abfrage Schalter
  {
   SwitchstateBuffer = false;
   Advertising = false;
   Directprint = false;
   lcd.noCursor();
   lcd.noBlink();

   SelectedMsg = SelectedMsg + 1;
   if (SelectedMsg >  EEPromBanks - 1 )
     {
     SelectedMsg = 0;
     }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Bank: " + String(SelectedMsg) + " selected");
    delay(10);
    value = 50000;

    while (digitalRead(SwitchPin) == 0)
    {
     delay(1);
     if (value > 0) {value = value - 1;};

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

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

Ab jetzt ist das Menü um 3 weitere Optionen reicher: ADVERT, ADSEC, und ADMSG. Diese zeigt unser Menü unter „Other“ an:


22 %
BlueTerm
connected: Tobis Display
------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

Other: ADVERT,ADSEC,ADMSG,ECHO

Type Cmd and press Enter
ADMsgs: (max 12)
Value 2 set.
ADverDelay: (max 255 Sec)
Value 5 set.
Advertising ON.

Als Handyapp empfehle ich zur Bedienung die App „BlueTerm“ die es kostenlos aus dem Google Store zu laden gibt. Um nun die Nachrichten nacheinander automatisch anzeigen zu lassen, ist wie folgt vorzugehen. Als erstes setzen wir mit dem Befehl ADMSG die Anzahl an Nachrichten von 0 zählend, die nacheinander angezeigt werden sollen. Beispiel: Wir haben in den Bänken 0-2 insges. 3 Nachrichten gespeichert, die angezeigt werden sollen. Dann geben wir nach dem Befehl ADMSG die Zahl 4 ein. Als Quittierung bekommen wir die Ausgabe: „Value 2 set“. Danach geben wir den Befehl „ADSEC“ ein gefolgt von einer Zahl zwischen 1 und 255. Dabei steht die eingegebene Zahl für die Anzahl an Sekunden, bis die angezeigte aktuelle Nachricht durch die nächste konfigurierte Nachricht ersetzt wird. Als Bespiel geben wir hier „5“ ein. Als Quittierung bekommen wir auch hier wieder die Ausgabe: „Value 5 set“. Als letzten Schritt geben wir nun den Befehl „ADVERT“ ein. Nach der Quittierung des Befehles durch „Advertising On“ wird nun immer im Wechsel automatisch die Nachrichten aus den Bänken 0-2 hintereinander anzeigt. Ausschalten lässt sich die Automatik durch erneute Eingabe des Befehles ADVERT oder durch einen druck auf den Nachrichtenwahltaster.

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