Im heutigen ersten Teil des mehrteiligen Blogs bauen wir uns ein frei konfigurierbares Bluetooth LCD-Display. Über ein serielles Bluetooth -Profil stellen wir dem Bluetooth-Client ein ASCII Menü zur Verfügung, über das das LCD-Display komplett steuerbar ist. Des Weiteren werden wir über die Blogteile hinweg das Menü erweitern und bauen neue (Komfort) Funktionen ein. Im ersten Teil der Reihe bauen wir das Auswahlmenü auf, und implementieren eine direkte Seriell zu Display Eingabe, die die getippten Buchstaben Zeichen für Zeichen direkt sichtbar auf unser Display ausgibt. Dabei verwenden wir das Steuerzeichen „Enter“ zum Wechsel der Displayzeilen (1-4) und zum Abschluss der Eingabesequenz, als auch das Steuerzeichen „Backspace“ zum Löschen von vorherigen Falscheingaben. Die Funktion werden wir im Menü „Direct Print“ nennen. Zum Aufbau der Bluetooth Verbindung kann zum einen ein Bluetooth Adapter am PC in Verbindung mit einem Terminalprogramm wie „Putty“ genutzt werden, als auch eine kompatible Handy APP, die ein serielles Bluetooth Profil unterstützt. Zur Übertragung der Seriellen Schnittstelle auf den Bluetooth Standard verwenden wir das bekannte Bluetooth Modul HC 05.

Wir schauen uns die Teileliste unseres Projektes an:

1x Bluetooth HC-05 Modul

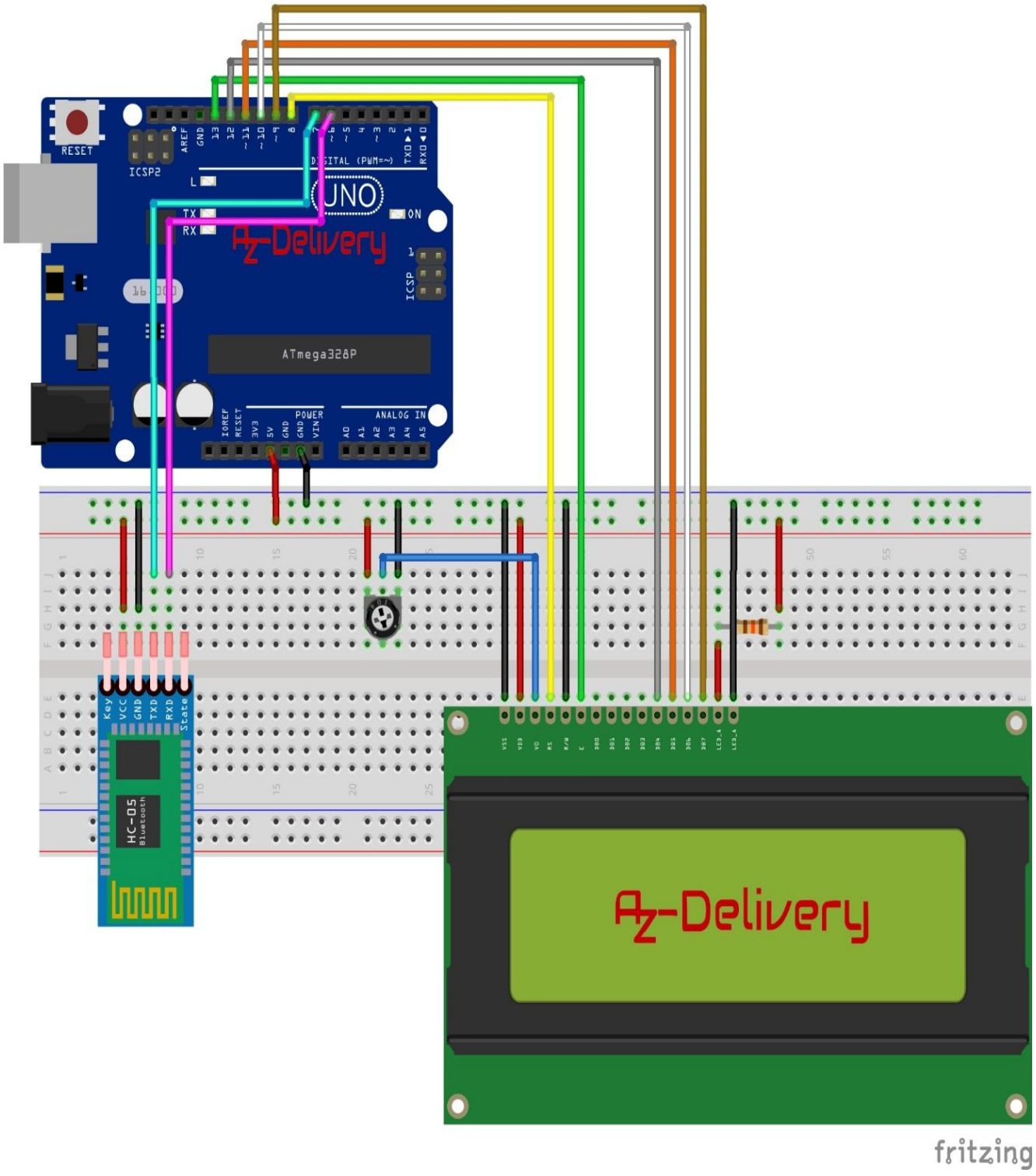
1x HD44780 LCD Display 4x20

1x Trimmer Potentiometer Max. Widerstand 10kΩ1

1x 130Ω Widerstand Toleranz ±5%

1x Uno R3 Board Typ UNO R3 ATmega328P

Es werden nicht viele Teile für dieses Projekt benötigt, auch hält sich die Beschaltung der einzelnen Bauteile in Grenzen.


RESET
ICSP2
UNO
TX
RX
ON
ICSP
ATmega328P
POWER
ANALOG IN
HC-05
Bluetooth
Key
VCC
GND
TXD
RXD
State
AZ-Delivery
fritzing

Wir laden auf unseren Arduino UNO folgenden Code hoch:

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>

#define MaxInputBufferSize 20 // maximal 255 Zeichen anpassen an vlcdr
#define rLcdChr 20
#define LcdRows 4
#define interval 1000

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

//variables

//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
int  MnuState = 0;
byte Ccount = 0;                          //Number received Chars
byte Inptype = 0;
boolean StrInput = false;
boolean NumberInput = false;
boolean DataInput = false;
boolean EnterInput = false;
byte MenueSelection = 0;
//Give Debug Informations over serial Interface
boolean DebugMode = false;
boolean EchoMode = true;
//Display Management
boolean Directprint = false;
byte DirectprintROW = 0;
byte DirectprintLine = 0;

boolean RefreshDisplay = false;

void setup()
{
  lcd.begin(rLcdChr, LcdRows);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("    Bluetooth   ");
  lcd.setCursor(0, 1);
  lcd.print("     Display    ");
  mySerial.begin(9600);
  lcd.setCursor(0, 0);
  mySerial.flush();
}

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 == "D") && (MnuState == 0) && (Inptype == 2))       { MenueSelection = 15;}
   switch (MenueSelection)
    {
       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;
      }
      default:
      {
        mySerial.println("-------Smart Bluetooth Display 1.0------");
        mySerial.println("D - Direct Print");
        mySerial.println("----------------------------------------");
        mySerial.println("Type Cmd and press Enter");
        mySerial.flush();
        MnuState = 0;
        value = 0;
        Sbuffer = "";
      }
    }
 } // Eingabe erkannt
// 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();
```

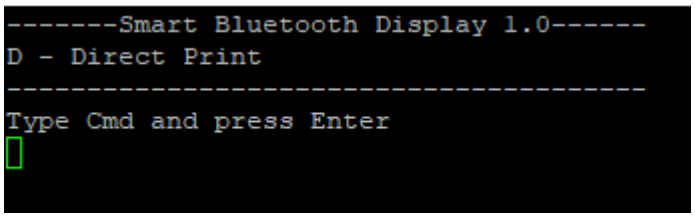
```
                } else {
                lcd.cursor();
                lcd.blink();
                }

           }
          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 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 ?bernommen, falls ben?tigt.
      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 ;
```

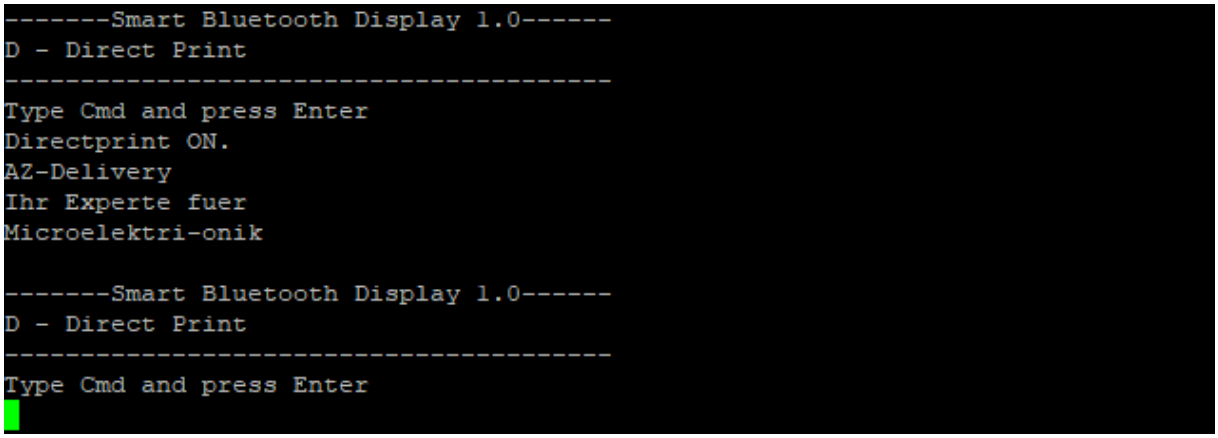
```
        b = b * 10;
        c = Cbuffer[2] - 48 ;
        value = a + b + c;
        }
      if (Ccount == 4) {
        a = Cbuffer[0] - 48 ;
        a = a * 1000;
        b = Cbuffer[1] - 48 ;
        b = b * 100;
        c = Cbuffer[2] - 48 ;
        c = c * 10;
        d = Cbuffer[3] - 48 ;
        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 loop()

{
SerialcommandProcessor();
}

//
##########################################################################################################
### //
```

Nachdem wir den Code hochgeladen haben, verbinden wir uns nun über das Bluetooth Modul und mit einem Seriellen Bluetooth Profil auf unseren Arduino. Wir bekommen folgendes Menü nach Betätigung der „Enter“ Taste:



Wir tippen jetzt den Buchstaben „D“ und drücken danach „Enter“:



Es folgt eine Bestätigung das „Directprint“ eingeschaltet wurde. Alles was wir nun tippen, wird direkt und ohne Verzögerung sowohl im LCD Display angezeigt als auch auf der Seriellen Schnittstelle quittiert. Wir „sehen“ also im Terminal, was wir schreiben und können auch Tippfehler mit „Backspace“ korrigieren.

Ein wichtiger Hinweis: Für diese in diesem Teil geschilderte Funktion des Displays mag der Code recht komplex erscheinen. Die reine Funktion wäre auch mit einfacherem Code zu realisieren. Auch sind bereits schon einige Teile des Codes für folgende Teile der Reihe optimiert bzw. vorbereitet worden.

Einzelne Abschnitte des Codes erklären u.u. sich erst in den folgenden Teilen

Nun wünsche ich euch viel Spaß beim Nachbauen und bis zum nächsten Mal.