Nano-Board mit dem Bluetooth-Modul HC-05 steuern

In der Einführung zum HC-05 (http://www.forum.g-heinrichs.de/viewtopic.php?f=16&t=133) haben wir schon ausführlich dargelegt, wie man die App "Bluetooth Electronics" mit dem Bluetooth-Modul kommunizieren lässt. Insbesondere haben Sie dort schon gelernt, wie man das Koppeln und Verbinden durchführt sowie ein Steuerelement auf ein Panel legt und damit arbeitet.

In diesem Beitrag wollen wir genauer darauf eingehen,

- wie man solche Steuerelemente konfiguriert,
- in welcher Form die Daten von der App gesendet werden,
- wie dazu passende Empfangsprogramme für das nano-Board aussehen können.

Stellvertretend für die in großer Zahl angebotenen Steuerelemente wollen wir den "Switch" und den "Slider" behandeln. Bei dem Switch handelt es sich um einen Schalter; im Gegensatz zu "Button" (Knopf), der schon in der oben erwähnten Einführung vorgestellt wurde, behält dieser den eingestellten Zustand auch beim Loslassen bei. Der Slider ähnelt einem Scrollbalken; mit ihm lassen sich Zahlenwerte einstellen und übertragen. Damit können z. B. eine LED gedimmt oder die Geschwindigkeit und Drehrichtung eines Motors gesteuert werden.

1. Switches

Auf ein leeres Panel legen wir die drei verschiedenen Schalter aus Abb. 1. Dazu aktivieren wir in der Auswahlliste den Steuerelemente-Typ "Switches" und ziehen die gewünschten Knöpfe auf das Panel.

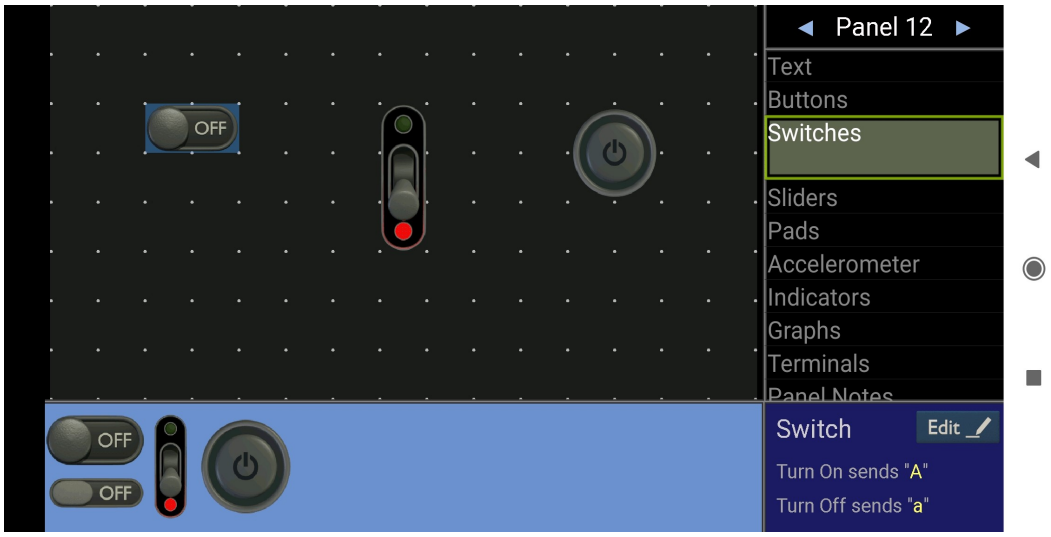

Abbildung 1

Zunächst kümmern wir uns um den Schalter oben links: Beim Einschalten wird der ASCII-Code für den Buchstaben "A", also die Zahl 65, ausgesendet. Dies können wir den Erläuterungen in dem

rechten unteren Editier-Feld entnehmen. Beim Ausschalten wird entsprechend der ASCII-Code von “a” gesendet. Den Schalterzustand können wir auch am Steuerelement selbst erkennen: Hier lesen wir je nach Zustand den Text “OFF” bzw. “On” ab.

Mit diesem Schalter wollen wir nun die Test-LED an PortB.5 ein- und ausschalten. Das zugehörige Programm ist einfach und sieht fast so aus wie das Programm für den Button in der oben erwähnten Einführung; lediglich der Zeichencode wurde an die Vorgabe für unseren Switch angepasst.

```
$regfile = "m328pdef.dat"
$crystal = 16000000
$framesize = 32
$swstack = 32
$hwstack = 64
$baud = 9600

Dim Zeichencode As Byte

Do
  Inputbin Zeichencode
  If Zeichencode = Asc( "A") Then Portb.5 = 1 Else Portb.5 = 0
Loop
```

Programm 1

Beim Austesten können wir feststellen: Einmal eingestellt, behalten sowohl der Schalter auf dem Panel als auch die LED ihren Zustand bei, bis der Schalter erneut umgestellt wird.

Wenden wir uns dem mittleren Schalter zu. Sein Editierfeld zeigt: Standardmäßig werden hier nicht die Codes von “A” und “a” gesendet, sondern die Codes von “C” und “c”. Die App schlägt hier andere Codes vor, damit je nach Code auch andere Steuerbefehle vom Mikrocontroller durchgeführt werden können.

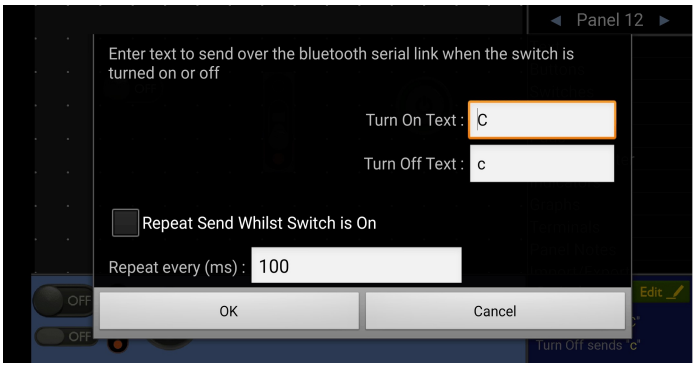

Abbildung 2

Hier allerdings wollen wir mit diesem Schalter auch wieder nur unsere LED an PortB.5 ein- und ausschalten. Wir könnten dazu unser obiges Programm anpassen, indem wir das Zeichen “A” durch das Zeichen “C” ersetzen. Eine andere Möglichkeit besteht darin, den Steuercode unseres Schalters zu ändern. Dazu betätigen wir nun die Schaltfläche **Edit** im Editierfeld. Wir erblicken zwei Felder, in denen wir die Codes “C” und “c” in “A” und “a” abändern können. In Wirklichkeit benutzen wir den Code “a” in unserem Programm 1 gar nicht: so wie unser Programm geschrieben worden ist, würde es ausreichen, hierfür irgendeinen anderen Buchstaben außer “A” einzutragen. Es ist aber sinnvoll und deswegen auch üblich, das Einschalten durch einen Großbuchstaben und das Ausschalten durch den zugehörigen Kleinbuchstaben zu kennzeichnen.

Im Übrigen sehen wir, dass es kurze Kommentare als Hilfestellung für die Einstellungen gibt. In diesem Fall können wir dieser Hilfestellung auch entnehmen, dass man dafür sorgen kann, dass die Steuercodes nicht beim Betätigen des Schalters, sondern vielmehr dauerhaft gesendet werden, solange der Schalter eingeschaltet ist; dazu müsste nur das Kästchen “Repeat Send...” aktiviert werden. Das unterlassen wir aber. Stattdessen betätigen wir die OK-Schaltfläche und gelangen dadurch wieder in das Panel-Fenster aus Abb. 1. Im Editierfeld unten rechts sehen wir, dass die Steuercodes unseres Schalters jetzt wie gewünscht “A” und “a” lauten.

Auch diesen Schalter sollten Sie mit dem Programm 1 austesten. Wenn Sie mögen, können Sie anschließend zur Übung auch auf ähnliche Weise mit dem rechten Schalter verfahren.

Nun wollen wir zwei verschiedene LEDs mit zwei Schaltern steuern. Dazu schließen wir zwei LEDs an die Pins D.2 und D.3 (mit Vorwiderständen) an. Einfacher ist es, wenn man ein LED-Bar (wie unter http://www.forum.g-heinrichs.de/viewtopic.php?f=16&t=127 vorgestellt) anschließt; hier sind die benötigten Vorwiderstände bereits auf dem Modul eingebaut.

Hierzu wählen wir ein neues Panel und platzieren darauf zwei Schalter wie in Abb. 3. Wenn Sie mögen, können Sie ihnen mit zwei Text-Elementen die Namen D.3 und D.2 geben.

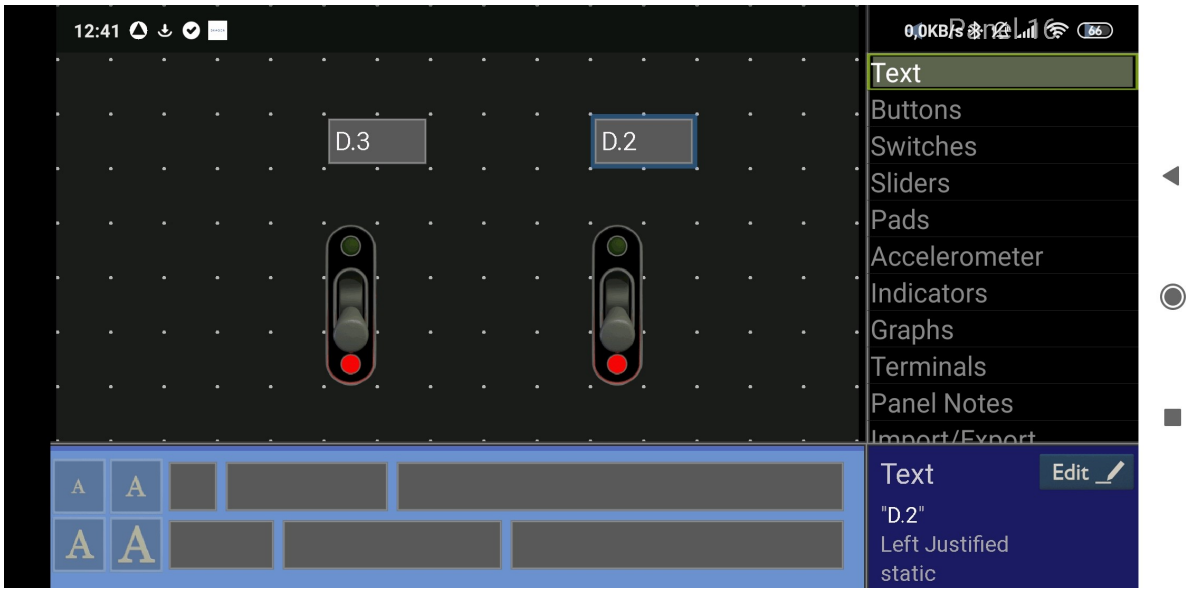

Abbildung 3

Damit der Mikrocontroller auf die beiden Schalter unterschiedlich reagieren kann, müssen wir ihnen unterschiedliche Steuercodes geben: Der linke Schalter soll die Codes “A” und “a”, der rechte die Codes “B” und “b” zum Ein- und Ausschalten erhalten. Führen Sie dazu die entsprechenden Konfigurationen durch, wie sie oben beschrieben worden sind.

Ein passendes Programm für das Nano-Board kann z. B. so aussehen:

```
$regfile = "m328pdef.dat"
$crystal = 16000000
$framesize = 32
$swstack = 32
$hwstack = 64
$baud = 9600

'************************************************************
'******************* Deklarationen ***********************

Dim Zeichencode As Byte
Dim Zeichen As String * 1

'************************************************************
'****************** Initialisierung *********************

Ddrd.2 = 1
Ddrd.3 = 1

'************************************************************
'******************** Hauptprogramm *********************

Do
  Inputbin Zeichencode
  Zeichen = Chr(zeichencode)
  Select Case Zeichen
    Case "A" : Portd.3 = 1
    Case "a" : Portd.3 = 0
    Case "B" : Portd.2 = 1
    Case "b" : Portd.2 = 0
  End Select
Loop
```

Programm 2

Bitte beachten Sie dabei: Wenn Sie den LED-Bar benutzen, müssen die Bits zum Ein- und Ausschalten invertiert werden.

2. Slider

Mit Slidern umzugehen ist schon etwas komplizierter: Neben dem Code, mit dem ein Slider identifiziert wird, muss auch der Positionswert des Schiebe-Knopfs übertragen werden. In Abb. 4 sehen wir im Editierfeld des Sliders, auf welche Weise dies geschieht.

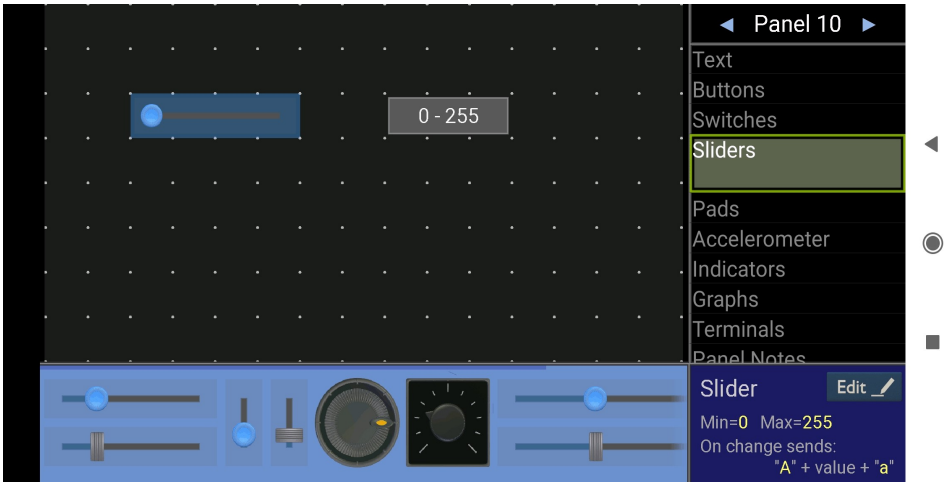

Abbildung 4

Wenn der Schiebe-Knopf verschoben wird, wird zuerst das Zeichen "A" gesendet, dann der Wert für die (neue) aktuelle Position des Knopfes und zum Abschluss das Zeichen "a". Wie gewohnt lassen sich das Start- und das Endzeichen durch den Benutzer ändern; die gilt auch für den Minimal- und den Maximalwert der Knopfposition. In diesem Fall ist der Wert am linken Anschlag des Knopfes 0 und am rechten Anschlag ist er 255.

Schauen wir uns einmal das Signal an, welches vom Bluetooth-Modul an den Mikrocontroller gegeben wird, wenn der Slider zunächst auf die Position 9 und dann auf die Position 15 geschoben wird (Abb. 5).

Abbildung 5

Offensichtlich werden hier die Werte ***nicht als eine einzige Zahl*** übertragen; vielmehr werden die einzelnen Ziffern der Reihe nach als Zeichen, genauer gesagt in Form von deren ASCII-Codes übertragen. Für den Fall der Position 15 bedeutet das: Hier wird die Zeichenkette "A15a" übertragen.

Eine Folgerung ist: Wie viele Zeichen insgesamt für eine Positionsübertragung erforderlich ist, hängt vom Positionswert ab. Im ersten Fall sind es 3 Zeichen, im zweiten 4 Zeichen. Damit ist auch klar, welche Bedeutung das Endzeichen hier hat: An diesem soll der Mikrocontroller erkennen können, wann die empfangene Ziffernfolge ihr Ende erreicht hat.

Damit ist auch klar, wie der Mikrocontroller mit den empfangenen Bytes umgehen muss: Zunächst muss er bei jedem Byte nachprüfen, ob es sich um den ASCII-Code für das Zeichen "A" handelt. Ist dies der Fall, dann wird das nächste Byte entgegen genommen. Handelt es sich hierbei nicht um den ASCII-Code für "a", dann muss sich das Programm dies als ASCII-Code der ersten Ziffer merken; so geht es in ähnlicher Weise weiter, bis der Mikrocontroller schließlich das Endzeichen empfängt. Der Mikrocontroller setzt die Zeichen zu einer Zeichenkette zusammen; aus dieser Zeichenkette kann er mit der val-Funktion die entsprechende Zahl erzeugen; damit kennt der Mikrocontroller jetzt den Zahlenwert der Knopfposition.

Mancher fragt sich vielleicht an dieser Stelle, warum hier nicht nach dem Startzeichen ein einziges Byte gesendet wird, dass direkt den Wert der Knopf-Position angibt. Würde man so vorgehen, dann wäre man aber auf Zahlen beschränkt, die kleiner als 256 sind; außerdem könnte man auf diese Weise nicht so einfach negative Zahlen oder sogar Dezimalzahlen übertragen. Der Weg über die Zeichenkette mit Start- und Endzeichen liefert eine flexible Übertragungsmöglichkeit. Die Flexibilität ist nicht kostenlos: Den Preis bezahlen wir, wenn wir dafür sorgen müssen, dass der Mikrocontroller die empfangenen Zeichen auswerten muss.

Wir wollen nun die Position des Sliders in binärer Darstellung mit Hilfe von LEDs anzeigen. Da als Maximalwert hier 255 gewählt worden ist, können wir den Positionswert in einer Variablen vom

Type Byte abspeichern. Um diesen Wert dann anzuzeigen, benötigen wir 8 LEDs; auch hier bietet es sich wieder an, ein LED-Array zu benutzen, z. B. den schon erwähnten LED-Bar.

In der folgenden Box ist ein passendes Programm angegeben. Bei diesem Programm wird das "virtuelle PortE" (vgl. http://www.forum.g-heinrichs.de/viewtopic.php?f=16&t=127) des nano-Boards benutzt.

Bitte beachten Sie dabei: Wenn Sie den LED-Bar benutzen, müssen Sie die Bits der Variablen "Zahl" invertieren.

```
$regfile = "m328pdef.dat"
$crystal = 16000000
$framesize = 32
$swstack = 32
$hwstack = 64
$baud = 9600

$lib "vPortE.LIB"
$external Porte
Declare Sub Porte(byval X As Byte)

'*********************************************************
'****************** Deklarationen ************************

Dim Zeichencode As Byte
Dim Zahl As Byte
Dim Zk As String * 3
Declare Sub Warte_auf_startcode

'*********************************************************
'****************** Initialisierung **********************



'*********************************************************
'******************** Hauptprogramm **********************

Do
  Warte_auf_startcode
  Inputbin Zeichencode                                    'empfängt erste Ziffer
  If Zeichencode <> Asc( "a") Then
    Zk = Chr(zeichencode)
    Inputbin Zeichencode                                  'empfängt zweite Ziffer
    If Zeichencode <> Asc( "a") Then
      Zk = Zk + Chr(zeichencode)
      Inputbin Zeichencode                                'empfängt dritte Ziffer
      If Zeichencode <> Asc( "a") Then
        Zk = Zk + Chr(zeichencode)
      End If
    End If
  End If
  Zahl = Val(zk)
  Porte Zahl                                              'Ausgabe auf dem LED-Array
Loop

'*********************************************************
'****************** Unterprogramme ***********************

Sub Warte_auf_startcode
  Do
    Inputbin Zeichencode
  Loop Until Zeichencode = Asc( "A")
End Sub
```

Programm 3

Nun ist es ein Leichtes, mit Hilfe eines Sliders eine LED zu dimmen. Dazu schließen wir unsere LED (nebst Vorwiderstand) an PortD.5 an; dies ist der PWM-Ausgang OC0B vom Timer0, einem 8-Bit-Timer. Durch die Konfiguration

```
Tccr0a = &B00100011
Tccr0b = &B00001010
Ocr0a = 100                'Maximalwert 100 -> 50 us (Periodendauer)
Ocr0b = 0                  'Vergleichswert beim Start des Programms
```

erzeugt der Timer0 beim Vergleichswert `Ocr0b = 10` periodisch Pulse mit der Periodendauer 50 us und der Pulsweite 10/100 · 50 us = 5 us. Wenn wir jetzt die Zeile

```
Zahl = val(Zk)
```

durch

```
Ocr0b = val(Zk)
```

ersetzen, dann lässt sich die Pulsweite durch den Slider kontinuierlich von 0 bis zur Periodendauer verändern. Der Dimmer ist fertig!

Nicht vergessen:

1. PortD.5 als Ausgang konfigurieren!
2. Slider-Werte konfigurieren: Min = 0, Max = 100

Auf dieselbe Weise können wir auch einen Gleichstrom-Motor steuern; da die Ströme, welche der Mikrocontroller abgeben kann, nicht zum Antreiben des Motors ausreichen, muss ein Treiber benutzt werden, z. B. ein L298-Modul. Ein solches Modul besitzt neben dem eigentlichen Treiber-IC u. A. einen Kühlkörper, Anschlussklemmen und eine galvanische Trennung der Eingänge.

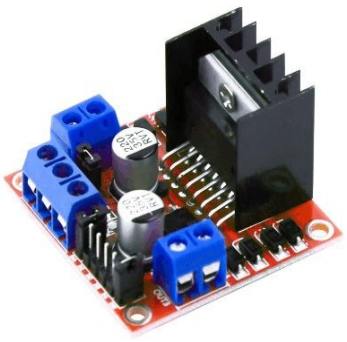

Abbildung 6

Der Eingang des Moduls wird mit PortD.5 bzw. Masse verbunden, der Ausgang mit dem Motor. Für den Motor benutzen wir allerdings eine größere Periodendauer; dazu stellen wir den Prescale-Wert von Timer0 mit

```
Tccr0b = &B00001101
```

auf 1024 ein.