Bluething Board

Datenübertragung per Bluetooth zwischen Arduino Nano und Smartphone, PC oder einem anderen Arduino Nano

Mit dem ‚Franzis Bluething Board' (Best.-Nr. CQ-14 48 59) kann man mit geringem zeitlichen und finanziellen Aufwand Sensor- oder sonstige Daten oder Schaltbefehle über Bluetooth – also über eine Kurzstrecken-Funkverbindung – übertragen. Das Bluething Board fasst einen Arduino Nano und ein HC-05-Bluetooth-Modul auf einer kleinen Platine zusammen. Dadurch kann man sehr schnell einen Arduino Sketch (oder ein Programm in einer anderen Programmiersprache für den eingebauten ATmega328-Mikrocontroller) erstellen, mit dessen Hilfe man einerseits das Bluetooth-Modul konfigurieren und andererseits mit einem Bluetooth-Partner kommunizieren kann. Im zweiten Fall realisiert das integrierte Bluetooth-Modul den Sender und Empfänger für eine virtuelle serielle Verbindung über Bluetooth. Dieser Artikel beschreibt auf Basis von mitgelieferten Beispielprogrammen, dass die Nutzung des Boards vielfältige Möglichkeiten bietet, aber gleichzeitig ein Einstieg in die Nutzung mit keinen bis geringen Vorkenntnissen möglich ist.

Bluething Board

Das Bluething Board ist eine flache, gerade einmal 2,1 x 7,6 cm kleine bestückte Platine, die verschiedene Komponenten sehr sinnvoll zusammenfasst (Bild 1). Die zwei Hauptkomponenten des Boards sind ein Mikrocontroller (ATmega328 mit 32 kB Flash-Speicher, 2 kB SRAM-Speicher, 1 kB EEPROM-Speicher, 16 MHz Taktrate) und ein HC-05-Bluetooth-Funkmodul. Durch ein USB-zu-Seriell-IC (CH340G), einen Spannungsregler, der den Betrieb auch an Spannungen von 7 bis 12 Volt ermöglicht, Leuchtdioden und Widerstände und einen bereits enthaltenen Bootloader, der die Programmierung des Mikrocontrollers über USB ermöglicht, wird insgesamt ein mit Arduino Nano kompatibles Modul im linken Teil des Boards realisiert. Den kompletten Schaltplan eines Arduino Nano findet man unter [1]. Im Zentrum des rechten Teils des Bluething Boards befindet sich ein HC-05-Bluetooth-Modul [2]. Damit dieses mit den Arduino-üblichen 5 Volt betrieben werden kann, wurde ein Spannungsregler verbaut und für die sichere serielle Kommunikation zwischen Arduino (5 V) und HC-05 (3,3 V) sorgt ein einfacher Levelshifter.

Die Verbindungen für eine serielle Verbindung vom PC und die Verbindungen vom Arduino zum Bluetooth-Modul sind auf der Platine bereits fest geschaltet.

Am Arduino bleiben noch etliche Pins für digitale oder analoge Ein- oder Ausgaben frei.

Wie bei fast allen Arduino-Boards gibt es eine eingebaute (gelbe) LED an Arduino-Pin 13 (Bild 2, neben der USB-Buchse). Außerdem gibt es im Arduino-Teil vier LEDs, die die Aktivitäten auf den seriellen Verbindungsleitungen anzeigen, und im HC-05-Teil zwei blaue LEDs, die den Status der Bluetooth-Verbindung angeben (Bild 2, Tabelle 1).

Die Verbindung zum PC erfordert ein USB-A-zu-Micro-USB-Kabel, das entweder schon vorhanden ist oder separat bestellt wird (Best.-Nr. CQ-12 00 51). (Andere Arduino Nanos haben oft eine Mini-USB-Buchse.) Durch die Zusammenfassung aller erforderlichen Komponenten auf einer Platine kann man ohne jegliches Löten oder Stecken sofort loslegen.

Zu beachten ist, dass es bei Bluetooth verschiedene Versionen gibt [3], die von den jeweiligen Kommunikationspartnern unterstützt werden müssen. Das verbaute HC-05-Bluetooth-Modul arbeitet mit der Bluetooth-Version 2.0+EDR. Damit ist eine Verbindung mit Android-Smartphones, PCs, Bluetooth-Sticks oder anderen Bluething-Boards möglich. Für eine Verbindung zu einem iOS-Gerät (iPhone, iPad) müsste man statt des Bluething Boards einen Arduino Nano (Best.-Nr. CQ-14 48 60) und zum Beispiel ein HM-10-Bluetooth-Modul kaufen, das auch die Bluetooth-Version 4.0 BLE unterstützt, die ein iPhone/iPad verwendet. Informationen zum Bluething Board findet man auch unter [4] und [5].

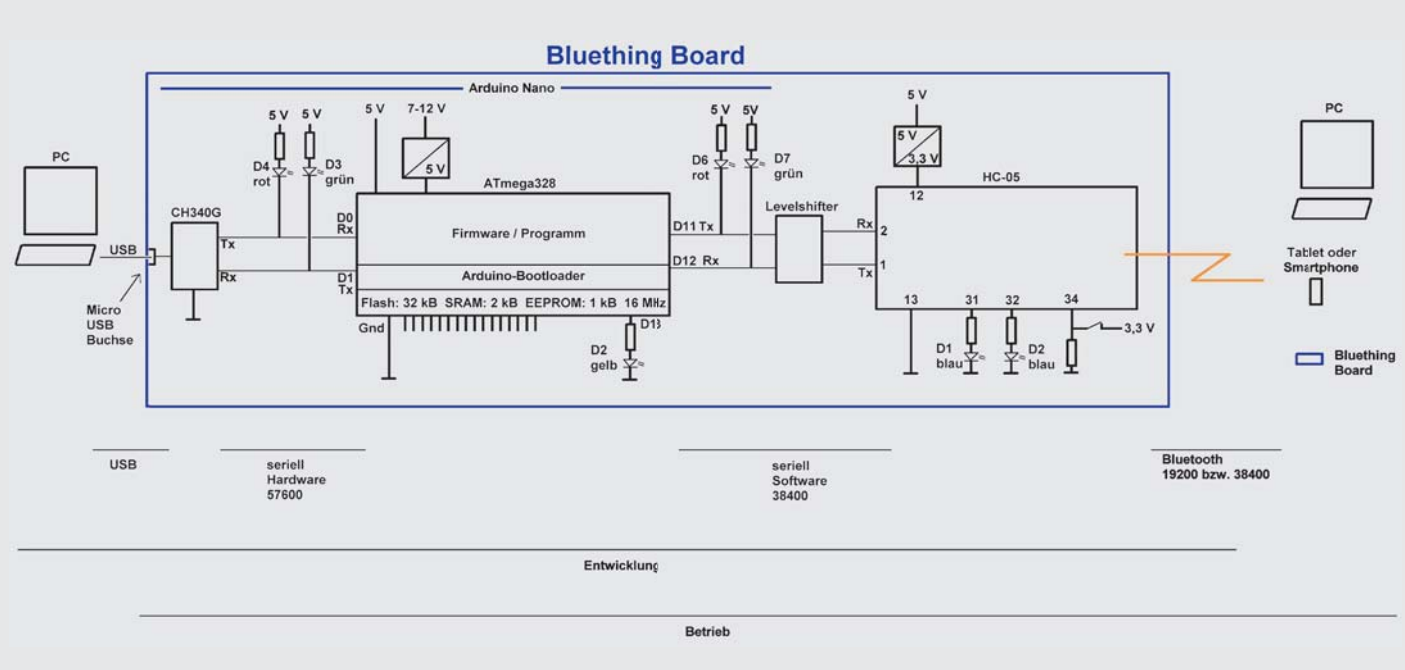


Bild 1: Bluething Board Blockbild

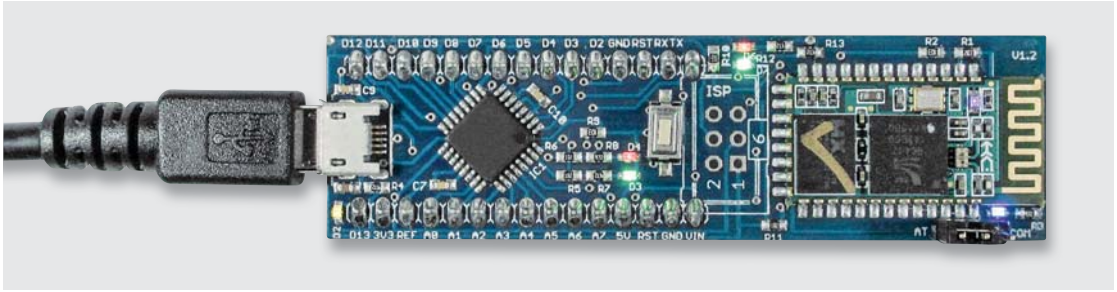


Bild 2: Bluething Board LEDs

Tabelle 1

LEDs auf dem Bluething Board

Pin	Funktion	Farbe	Bezeichnung auf Platine
Arduino: 0	Rx Hardwareserial	Rot	D4 neben Reset-Taster
Arduino: 1	Tx Hardwareserial	Grün	D3 neben Reset-Taster
Arduino: 11	Tx Softserial	Rot	D6 neben ISP
Arduino: 12	Rx Softserial	Grün	D7 neben ISP
Arduino: 13	Onboard-LED	Gelb	D2 neben USB-Buchse
HC-05	Blinken 1 Hz: sucht Blitzen 2 Hz: Pairing erfolgt Doppelblitzen 2 Hz: verbunden	Blau	Obere blaue LED
HC-05	Aus: Nicht verbunden An: Verbunden	Blau	Neben HC-05 Steckbrücke

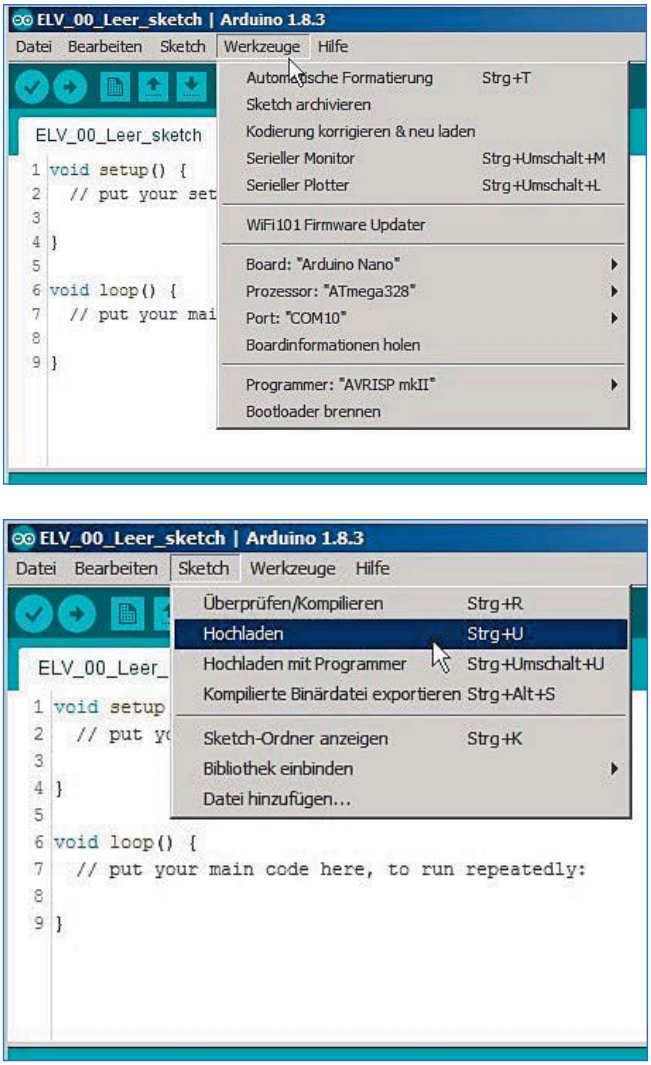


Bild 3: Einstellungen Arduino-Entwicklungsumgebung und Hochladen Sketch

Arduino-Teil

Der Arduino-Teil des Boards (linker Teil in Bild 1 und in Bild 2) kann wie ein ganz normaler Arduino Nano benutzt werden. Wenngleich über den integrierten Bootloader oder die vorhandenen ISP-Anschlüsse auch andere Programmierumgebungen/-sprachen verwendet werden können, bietet sich die Nutzung der Arduino-Entwicklungsumgebung/Sprache an und wird auch in den Beispielen benutzt. Eine ausführliche Einführung in die Arduino-Programmierung kann unter [6] heruntergeladen werden. Hier sollen nur die wesentlichen Aspekte für die Benutzung des Bluething Boards dargestellt werden.

Installation

Falls die Arduino-Entwicklungsumgebung nicht bereits installiert ist, muss sie von [7] heruntergeladen und installiert werden. Wichtig ist, dass bei der Installation oder, falls das nicht reibungslos klappen sollte, separat von [4], der Treiber für den USB-zu-Seriell-Umsetzer CH340G installiert wird, damit der Arduino Nano des Bluething Boards beim Verbinden mit dem PC richtig erkannt und unterstützt wird.

Nach erfolgreicher Installation der Arduino-Umgebung sieht man als Erstes ein Programm – in der Arduino-Welt „Sketch" genannt –, weil automatisch im Hintergrund einiges für den Benutzer dazugefügt wird. Dieser erste Leersketch ist bereits ein ausführbarer Sketch, der die zwei wesentlichen Blöcke eines Arduino-Sketches zeigt:

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Im Setup-Teil werden Dinge geschrieben, die einmalig am Anfang des Sketches ausgeführt werden sollen. Im Loop-Teil werden die Aktionen beschrieben, die in unendlicher Wiederholung durchgeführt werden sollen.

Übertragen eines Sketches auf den Arduino Nano

Um einen Sketch auf den Arduino Nano zu übertragen, müssen zunächst unter „Werkzeuge" die virtuelle serielle Schnittstelle (Port, siehe Windows-Gerätemanager) und das verwendete Arduino-Board eingestellt werden (Bild 3 oben). Dann wird im Menü unter „Sketch" die Option „Hochladen" gewählt

(Bild 3 unten). Der fehlerfreie Leersketch wird dadurch zunächst auf Fehler geprüft, dann in Maschinenbefehle umgewandelt und schließlich auf den ATmega328 geladen. Der Sketch wird auf dem Mikrocontroller sofort gestartet – in diesem Fall würde allerdings nichts Sichtbares passieren, weil es sich bei dem Leersketch nur um eine Art Skelett für eigene Sketche handelt.

Sketch Blinken

Ein erster Sketch, der eine Funktion ausführt, ist meistens ein Sketch, der eine LED zum Blinken bringt:

```
//Test-Sketch zum Überprüfen der Grundfunktionen des Moduls
//Dieser Sketch bringt lediglich eine LED auf der Platine zum Blinken
#define LedPin 13              // Gelbe LED links unten (neben USB-Buchse)

void setup() {
  pinMode(LedPin, OUTPUT);     // Pin13 als Ausgang setzen
}

void loop() {                  // Endlosschleife
  digitalWrite(LedPin, HIGH); // LED an
  delay(100);                  // warten
  digitalWrite(LedPin, LOW);  // LED aus
  delay(1000);                 // warten
}
```

Der Sketch fängt mit Kommentarzeilen an, die mit // gekennzeichnet werden. Danach wird durch die Präprozessor-Direktive #define festgelegt, dass überall im Sketch, wo LedPin steht, stattdessen 13 eingesetzt wird. In der Setup-Funktion wird der entsprechende Pin als Ausgangspin definiert. In der Schleifen-Funktion (loop) wird der entsprechende Pin mit digitalWrite auf HIGH (5 V) gesetzt, mit delay 100 ms gewartet, der LedPin auf LOW (0 V) geschaltet und wieder gewartet. Dadurch, dass das wiederholt geschieht, wird die angeschlossene LED zum Blinken gebracht. Hinter jeder Anweisung ist ein Semikolon (;) zu schreiben. Groß-/Kleinschreibung ist bei Arduino zu beachten!

Sketch Blinken mit Taster

Ein Sketch zum Abfragen eines Pins erweitert den Blink-Sketch. Die wichtige neue Funktion, die hier benutzt wird, heißt digitalRead und ermöglicht das Ermitteln des an einem Pin anliegenden Spannungspegels. Als Pinmode wird hier INPUT_PULLUP definiert, wodurch ein interner Pull-up-Widerstand geschaltet wird, der bei offenem Pin einen HIGH-Zustand erzeugt. Mit dem Schlüsselwort „if" wird eine Fallunterscheidung beschrieben, wodurch je nach Tasterzustand (gedrückt/nicht gedrückt) unterschiedlich lange Wartezeiten durchgeführt werden.

```
/*Test-Sketch zum Überprüfen der Grundfunktionen des Moduls
  Dieser Sketch bringt eine LED auf der Platine zum Blinken.
  Wenn der Taster (gegen Gnd) gedrückt ist, blinkt die LED schneller
*/
//#define LedPin 13             // Auf dem Nano gelbe LED links unten (neben USB-Buchse)
#define LedPin LED_BUILTIN      // Die Konstante LED_BUILTIN steht für die Onboard-LED. Fast immer D13
#define TASTER_PIN 10           // Pin für Taster
#define JA LOW

void setup() {
  pinMode(LedPin, OUTPUT);              // Pin 13 als Ausgang setzen
  pinMode(TASTER_PIN, INPUT_PULLUP);    // Pin 12 als Eingang mit Pull-up-Widerstand
}

void loop() {                                   // Endlosschleife
  int tasterGedrueckt = digitalRead(TASTER_PIN);
  digitalWrite(LedPin, HIGH);                   // LED an
  if (tasterGedrueckt == JA)
  {
    delay(100);                                 // Kurz (100 ms) warten
  }
  else
  {
    delay(1000);                                // Länger (1000 ms) warten
  }

  digitalWrite(LedPin, LOW);                    // LED aus
  delay(1000);                                  // 1 s warten
}
```

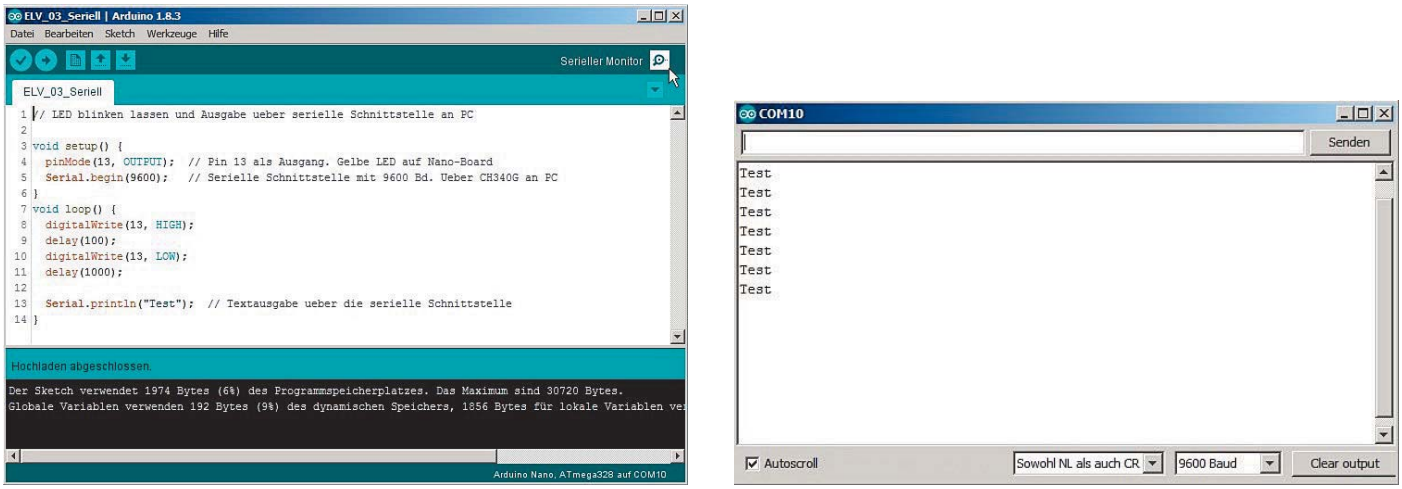


Bild 4: Serieller Monitor

Seriell, einfach

Der USB-zu-Seriell-Konverter CH340G ermöglicht nicht nur das Übertragen eines Sketches auf den Mikrocontroller, sondern auch eine serielle Kommunikation zwischen PC und Mikrocontroller während der Laufzeit. Im folgenden Sketch wird die serielle Schnittstelle (auch als UART bezeichnet) mit einer Geschwindigkeit von 9600 Baud geöffnet, und außer dem Blinken der LED wird in der Endlosschleife auch der Text „Test" permanent über die serielle Schnittstelle (an den PC) gesendet.

```
// LED blinken lassen und Ausgabe über serielle Schnittstelle an PC
void setup() {
  pinMode(13, OUTPUT);  // Pin 13 als Ausgang. Gelbe LED auf Nano-Board
  Serial.begin(9600);   // Serielle Schnittstelle mit 9600 Baud über CH340G an PC
}
void loop() {
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(1000);

  Serial.println("Test");  // Textausgabe über die serielle Schnittstelle
}
```

Um den vom Mikrocontroller gesendeten Text zu sehen, wird in der Arduino-Entwicklungsumgebung rechts oben der serielle Monitor geöffnet (Bild 4 links), in dem dann – ganz wichtig: nach Einstellen der korrekten Baudrate unten! – die ausgegebenen Texte angezeigt werden (Bild 4 rechts). Statt des eingebauten seriellen Monitors könnte man auch ein beliebiges Terminalprogramm wie zum Beispiel HTerm [8] benutzen. In jedem Fall muss auf beiden Seiten einer seriellen Kommunikation dieselbe Baudrate eingestellt sein!

Seriell mit Temperatur

Zum Ermitteln eines Temperaturwertes kann man einen Temperatursensor DS18B20 (Best.-Nr. CQ-10 27 83) gemäß Bild 5 (und Bild 10) an das Bluething Board anschließen und durch die Einbindung der entsprechenden Bibliotheken sehr leicht die Temperatur durch den Sensor ermitteln und per serieller Schnittstelle an den PC senden lassen (Bild 6).

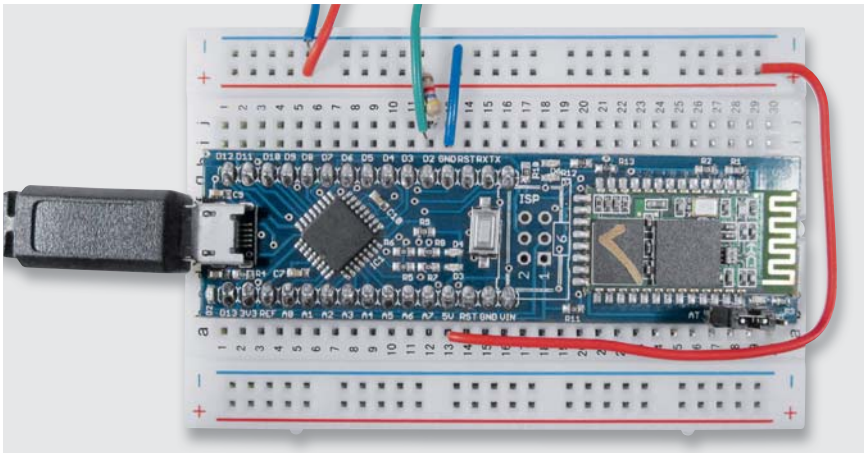
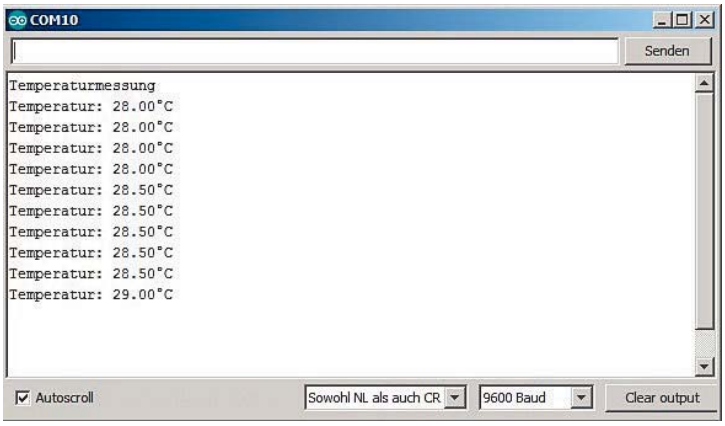


Bild 5: Bluething Board auf Steckbrett, Temperatursensor DS18B20 angeschlossen

Bild 6: Temperaturwerte im seriellen Monitor




```
// LED blinken lassen, Temperatur mit einem DS18B20-Temperatursensor messen und seriell ausgeben
// Sensor an Pin des Arduino angeschlossen (siehe define Sensorpin)
// Gnd von Sensor mit Gnd von Arduino verbunden
// Verbindung von 5 V des Arduino zu Vcc des Sensors
// Pull-up  4,7 kΩ zwischen Sensorpin und 5 V
// Messung von -55 °C bis +155 °C mit 0,5 °C Genauigkeit
// Wenn -127.00 angezeigt wird, dann ist der Sensor falsch angeschlossen

// -----  Fuer Temperaturmessung:  -----
#include <OneWire.h>                 // Bibliothek für OneWire einbinden
#include <DallasTemperature.h>       // Bibliothek für DS18B20 Temperatursensor einbinden
#define Sensorpin 2                  // Pin, an dem der DS18B20 angeschlossen ist. Hier D2
OneWire MeinBus(Sensorpin);          // OneWire Instanz initialisieren
DallasTemperature sensors(&MeinBus); // Dallas Temperatur-Library für Nutzung der OneWire Library vorbereiten
// ------------------------------------

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  Serial.println("Temperaturmessung");
  sensors.begin();                   // Dallas Temperatur-Library für Nutzung der oneWire Library vorbereiten
}

void loop() {
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  sensors.requestTemperatures();     // Temperatur abfragen
  delay(1000);                       // 1 s warten

  Serial.print("Temperatur: ");              // Serielle Ausgabe ohne CRLF
  Serial.print(sensors.getTempCByIndex(0) ); // Temperaturwert ausgeben
  Serial.println("°C");                      // Serielle Ausgabe mit CRLF
}
```

Ansprechen des Bluetooth-Moduls HC-05

Bis hierher wurde nur der Arduino-Teil des Bluething Boards verwendet. Nun kommt das Bluetooth-Modul ins Spiel. Das Bluetooth-Modul HC-05 (Bild 1 rechter Teil des Boards) kann in zwei Modi angesprochen werden. Im sogenannten AT-Modus werden Texte per serieller Verbindung an den HC-05 gesendet, die von diesem als (Konfigurations-)Befehle interpretiert werden [2]. Die Befehle beginnen immer mit AT. Im sogenannten Com-Modus werden seriell an den HC-05 gesendete Texte quasi durchgereicht und per Bluetooth gesendet. Zwischen den beiden Modi (AT/COM) kann durch Stecken des Jumpers auf der Platine oder per Software umgeschaltet werden.

Firmware: Nur senden

Der folgende Sketch konfiguriert zunächst das HC-05-Modul mit AT-Befehlen und sendet dann in einer Endlosschleife einen konstanten Text („ELV“) und die seit Sketch-Start vergangenen Millisekunden (millis()) per Bluetooth.

Da die serielle Schnittstelle (Hardware-UART) des Arduino-Mikrocontrollers bereits für das Übertragen des Sketches auf den Mikrocontroller und auch für die Kommunikation zum PC verwendet wird, wird hier – quasi als „Trick“ – eine weitere serielle Schnittstelle in Software abgebildet (SoftwareSerial). Wir haben also insgesamt drei serielle Schnittstellen: Zwischen PC und Arduino, zwischen Arduino und HC-05 und die Bluetooth-Funkverbindung (Bild 1, Bild 10). Für jede dieser seriellen Verbindungen muss die entsprechende Baudrate zwischen Sender und Empfänger übereinstimmen.

```
//Text von Arduino über das Bluetooth-Modul HC-05 per Bluetooth zum Handy oder PC senden
//Beim Aufspielen der Firmware muss der Jumper auf AT stehen. Außerdem darf das HC-05 Modul dabei nicht mit einem Handy
oder anderen Modul gekoppelt sein!

//Nachdem die Initialisierungs-AT-Befehle gesendet wurden und die gelbe LED leuchtet, kann ein Handy/PC mit dem Modul
gekoppelt werden. In dem Fall wechselt das Modul automatisch in den COM-Modus (unabhängig vom Jumper)
//Texte werden an Handy/PC gesendet

//Initialisieren der Softserial-Schnittstelle
#include <SoftwareSerial.h>             // Einbinden der Bibliothek für Software-UART
SoftwareSerial HC05(12, 11);            // Pin 11 und 12 für die Software-UART

void setup()  {
  pinMode(13, OUTPUT);                  // Pin13 als Ausgang setzen
  HC05.begin(38400);                    // Initialisieren mit 38400 Baud

  // HC-05 auf Defaultwere setzen -> Slave mode, Baudrate 38400, Passwort:1234, Device-Name: "hc01.com HC-05"
  HC05.println("AT+ORGL"); delay(500);

  // Lösche alle Devices aus der Pair-Liste
  HC05.println("AT+RMAAD"); delay(500);

  // Setze Name
  HC05.println("AT+NAME=ELVjournal"); delay(500);

  // Setze Pin auf 4321
  HC05.println("AT+PSWD=4321"); delay(500);

  // Setze Baudrate auf 19200
  HC05.println("AT+UART=19200,1,0"); delay(500);

  // Modul neustarten und eventuelle Verbindungen resetten
  HC05.println("AT+RESET"); delay(1000);

  // SPP Profile Lib initialisieren und disconnecten
  HC05.println("AT+INIT"); delay(500);
  HC05.println("AT+DISC"); delay(500);

  digitalWrite(13, HIGH);
  //  Konfiguration beendet!
  //  Die gelbe LED an Pin13 sollte jetzt konstant leuchten
  //  Der Bluetooth-Name des Moduls lautet: "ELVjournal" , das Passwort lautet: "4321"
  //  Das Modul kann jetzt mit einem Handy gekoppelt werden
}

void loop() {
  //Sende Zeichen zum Modul
  HC05.print("ELV  ");              // Sende String ohne CRLF an das Modul
  HC05.println(millis() / 1000);    // Sekunden seit Start des Programms werden per Bluetooth gesendet
  delay(1000);
}
```

Nachdem nun dieser Sketch den HC-05 konfiguriert (Name: ELVjournal, Baudrate: 19200 Baud, Passwort: 4321) und vorbereitet hat, kann ein Bluetooth-Empfänger – hier ein Smartphone – verbunden werden, um die gesendeten Daten zu empfangen.

Gemäß Bild 7 ganz links wird auf dem Android-Gerät in den Einstellungen Bluetooth eingeschaltet, durch Tippen auf das kleine Dreieck die zunächst leere Bluetooth-Pairingliste geöffnet und dort „weitere Einstellungen“ gewählt. Es werden sichtbare Bluetooth-Geräte in der Umgebung gesucht und angezeigt (Mitte). Nach Selektion des Bluething Boards (hier mit dem Namen „ELVjournal“) wird der Pin (hier 4321) eingegeben. Danach erscheint das Board in der Liste der Geräte mit Pairing (Bild 7 rechts).

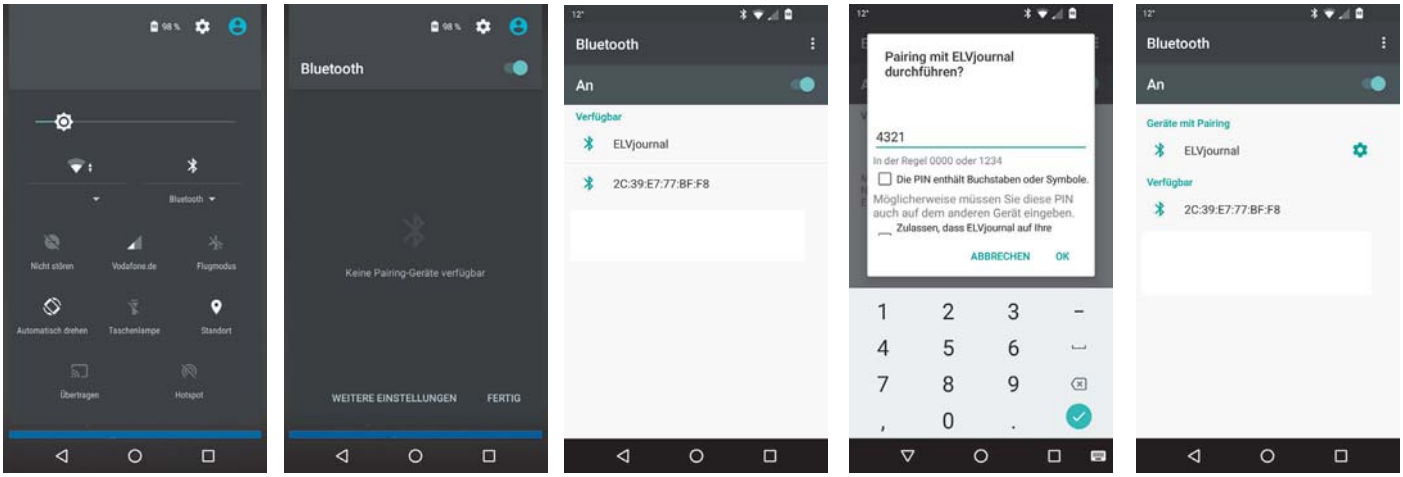


Bild 7: Pairing mit dem Bluething Board

Auf dem Android-Gerät wird nun eine Bluetooth-Kommunikations-App aus dem Google Play Store geladen und installiert. Als gutes Beispiel wird hier die App „Serial Bluetooth Terminal" verwendet [9].

In der App werden (über das Menü links oben, dann „Bluetooth Devices") die Geräte aus der Pairing-Liste angezeigt – hier „ELVjournal" (Bild 8 links). Über das Menü (links oben, dann Terminal) kommt man wieder zum eigentlichen Terminalfenster (Bild 8 rechts). Dort wird oben in der Menüzeile (rechts vom Text „Terminal") das kleine Symbol mit den zwei Steckern geklickt. Dadurch wird die Bluetooth-Verbindung hergestellt (connected) bzw. beim nochmaligen Betätigen wieder getrennt (disconnected).

Sobald die Verbindung hergestellt ist, sieht man die Ausgabe der Texte, die vom Bluething Board per Bluetooth gesendet werden (Bild 8 rechts)!

Firmware, komplett

Das komplette mitgelieferte Firmware-Beispiel – also der komplette Sketch für den Arduino – konfiguriert zunächst die drei seriellen Verbindungen und etliche Basiseinstellungen für das HC-05-Modul (siehe Sketch-Kommentare). Im seriellen Monitor auf dem PC wird jeweils der zum HC-05 gesendete AT-Befehl und die Antwort vom HC-05 angezeigt (Bild 9). Dies geschieht mithilfe der Funktion „auslesen", in der jeder AT-Befehl zunächst zu Kontroll- bzw. Protokollzwecken zum seriellen Monitor gesendet wird (Serial.println(Befehl)), dann wird der AT-Befehl zum HC-05-Modul gesendet (HC05.println(Befehl)) und abschließend werden alle vom HC-05-Modul zurückgesendeten Zeichen eingesammelt und als Gesamtzeichenkette zur Ausgabe an den seriellen Monitor mit der Funktion zurückgegeben.

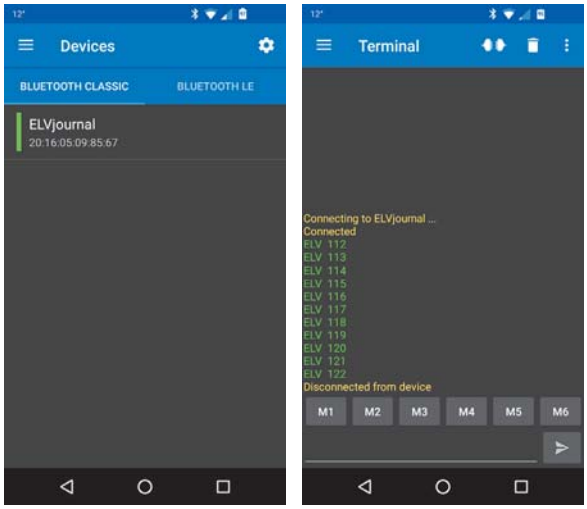


Bild 8: Terminal-App „Serial Bluetooth Terminal"

Die Funktion „auslesen":

```
String auslesen(String Befehl){
  char Zeichen;                 // Jedes empfangene Zeichen kommt kurzzeitig in diese Variable
  String result="";
  Serial.println(Befehl);       // Schreibe den übergebenen String auf den seriellen Monitor
  HC05.println(Befehl) ;        // Sende den übergebenen String an das Modul
  delay(500);
  while(HC05.available() > 0){  // Solange etwas empfangen wird, durchlaufe die Schleife
    Zeichen = HC05.read();      // Speichere das empfangene Zeichen in der Variablen "Zeichen"
    result.concat(Zeichen);     // Speichere die Antwort des Moduls
  }
  return result;                // Übergebe die Rückantwort des Moduls
```

Ein Aufruf der Funktion „auslesen" sieht dann beispielsweise so aus:

```
Text = auslesen("AT+ORGL");
  Serial.println(Text);
```

Hier wird die Funktion „auslesen" mit dem Befehl zum Herstellen der Werkseinstellungen (AT+ORGL) aufgerufen. Die Antwort vom HC-05 (hier OK) wird der Variablen „Text" zugewiesen und dann mit Serial.println an den seriellen Monitor auf dem PC gesendet (Bild 9).

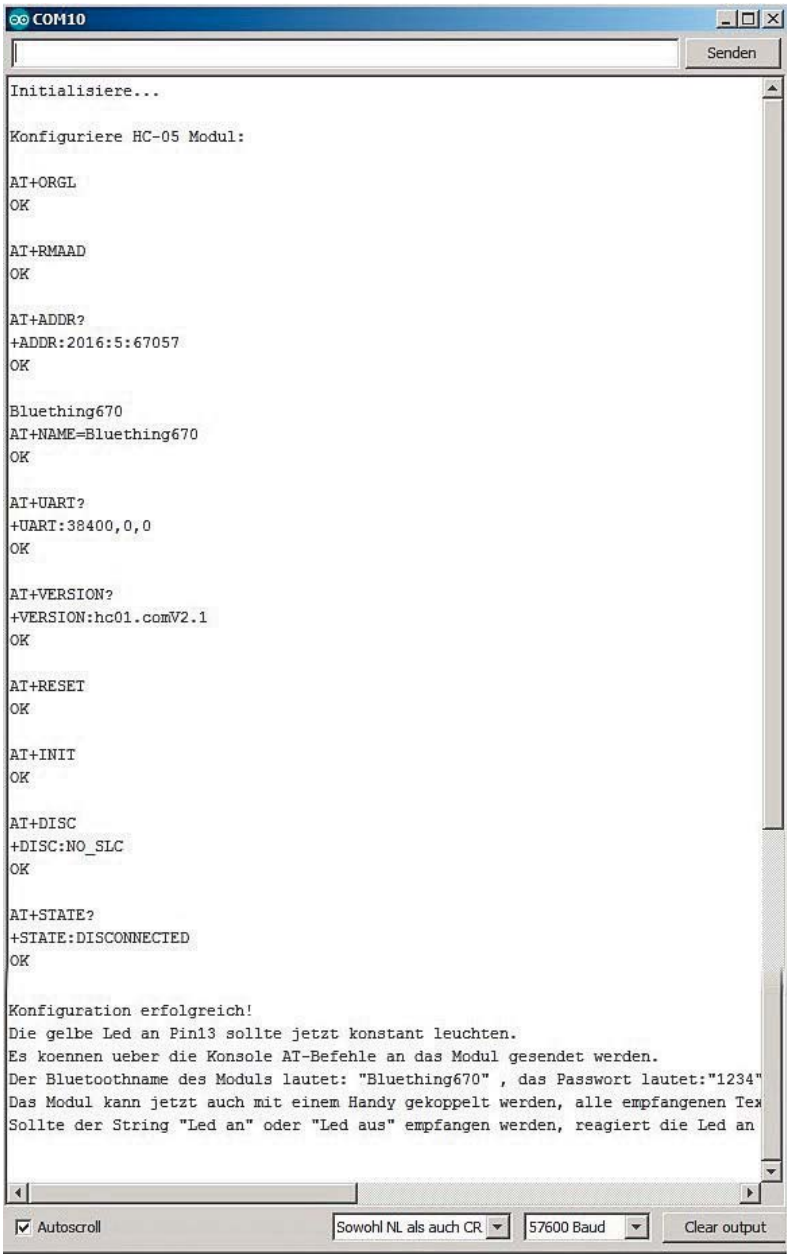


Bild 9: Bluetooth-Modul-Initialisierung im seriellen Monitor

Im Sketch wird jeweils kurz geprüft, ob die Antwort vom Modul fehlerfrei war, und andernfalls eine kurze Meldung auf dem seriellen Monitor ausgegeben.

Am Schluss des Setup-Teils wird eine kleine Gebrauchsanleitung auf dem seriellen Monitor ausgegeben (Bild 9).

Im Loop-Teil des Sketches wird eine bidirektionale Verbindung hergestellt, bei der Texte vom seriellen Monitor auf dem PC durch den Arduino durchgereicht und vom HC-05 per Bluetooth gesendet werden und umgekehrt vom HC-05 empfangene Texte durch den Arduino an den PC weitergereicht werden (Bild 10).

Weiterhin wird beim Durchreichen jedes Textes vom Arduino geprüft, ob es sich um einen Schaltbefehl für die gelbe LED handelt. Der Schaltbefehl wird ggf. ausgeführt.

Um das Szenario noch etwas spannender zu machen, wurde der Sketch für die Leser des ELV Journals dermaßen ergänzt, dass vom HC-05 empfangene Texte außerdem auf das Schlüsselwort „Temperatur" ausgewertet werden. Sobald dieses Schlüsselwort über Bluetooth empfangen wurde, wird die vom angeschlossenen Sensor ermittelte Temperatur per Bluetooth an den PC gesendet.

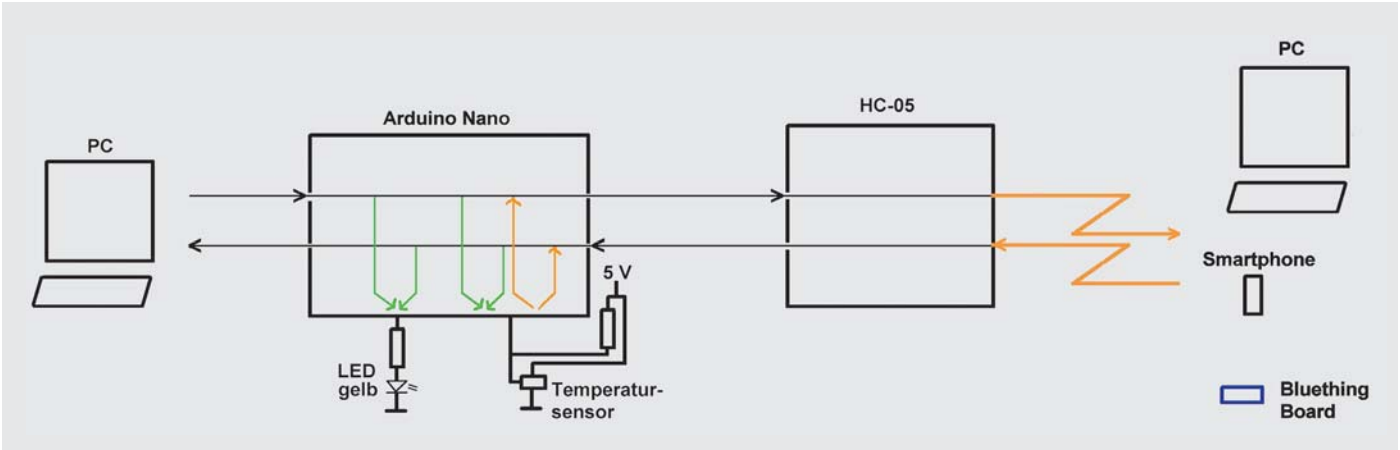


Bild 10: Schema des Beispielsketches

Der komplette Sketch sieht aus wie folgt:

```
//Firmware BlueThing V1.2b   MIT ERGÄNZUNGEN FÜR TEMPERATUR
//Beim Aufspielen der Firmware muss der Jumper auf AT stehen. Außerdem darf das HC-05-Modul dabei nicht mit einem Handy oder anderen Modul gekoppelt sein!
//Nachdem das Modul bereit ist, können über den seriellen Monitor AT-Befehle an das Modul gesendet werden.
//Die Antworten des Moduls werden dann auf dem seriellen Monitor angezeigt.
//Auch kann jetzt ein Handy mit dem Modul gekoppelt werden. In dem Fall wechselt das Modul automatisch in den COM-Modus (unabhängig vom Jumper).
//Jetzt werden alle über den seriellen Monitor gesendeten Befehle direkt ans Handy weitergeleitet.
//Empfängt das Modul von einem Handy Daten, werden diese auf dem seriellen Monitor angezeigt.
//Empfängt das Modul die Befehle "LED an" oder "LED aus" wird eine LED an Pin 13 entsprechend geschaltet.
//Empfängt das Modul den Befehl "Temperatur", wird der aktuelle Temperaturwert vom Arduino über das HC-05-Modul zum Handy gesendet.

String Name="Bluething";
String Text="";    // In diesem String werden Texte zwischengespeichert
char Zeichen;      // In dieser Variable werden Zeichen zwischengespeichert

#define txPin 11    // Virtueller Tx-Pin des Moduls zur Kommunikation mit dem HC-05
#define rxPin 12    // Virtueller Rx-Pin des Moduls zur Kommunikation mit dem HC-05
#define LedPin 13   // An diesem Pin hängt die gelbe LED unten links

//Initialisieren der virtuellen seriellen Schnittstelle
#include <SoftwareSerial.h>
SoftwareSerial HC05(rxPin, txPin);

// -----  Für Temperaturmessung:  -----
#include <OneWire.h>
#include <DallasTemperature.h>
#define Sensorpin 2                    // Pin, an dem der DS18B20 angeschlossen ist. Hier D2
OneWire MeinBus(Sensorpin);            // OneWire Instanz initialisieren
DallasTemperature sensors(&MeinBus);  // Dallas Temperatur-Library für Nutzung der OneWire Library vorbereiten
// ------------------------------------

void setup()  {
  pinMode(LedPin, OUTPUT); // Pin 13 als Ausgang setzen
  Serial.begin(57600);     // Baudrate für die Kommunikation mit dem seriellen Monitor auf dem PC

  Serial.println("Initialisiere...");
  Serial.println();

  Serial.println("Konfiguriere HC-05 Modul:");
  Serial.println();
  HC05.begin(38400); //Baudrate fuer Kommunikation zwische Arduino und HC05

  //Reset HC-05 -> Slave mode, Baudrate 38400, Passwort:1234, Device-Name: “hc01.com HC-05”
  Text = auslesen("AT+ORGL");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    Serial.println("Steht der Jumper auf AT?");
    Serial.println("Sind alle Bluetooth-Verbindungen zum Modul getrennt?");
    Serial.println("Wechsle in den COM-Modus...");
    Serial.println();
    return;
    //while(1);
  }


  //Lösche alle Devices aus der Pair-Liste
  Text = auslesen("AT+RMAAD");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }
```

```
  //Lese die Mac-Adresse aus und bilde daraus den Bluetooth-Namen
  Text = auslesen("AT+ADDR?");
  Serial.println(Text);
  Name.concat(Text.substring(Text.lastIndexOf(":")+1,Text.lastIndexOf(":")+4));
  Serial.println(Name);

  //Setze Bluetooth-Name auf "BluethingXXX"
  Text = auslesen("AT+NAME="+Name);
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }

  //Checke, ob die Baudrate auf 38400 gesetzt wurde
  Text = auslesen("AT+UART?");
  Serial.println(Text);
  if (!Text.startsWith("+UART:38400,0,0")) {
    Serial.println("Fehler!");
    while(1);
  }

  //Versionsnummer des Moduls auslesen - Keine Überprüfung, falls sich mal die Versionsnummer ändert ...
  Text = auslesen("AT+VERSION?");
  Serial.println(Text);

  //Modul neustarten und eventuelle Verbindungen resetten
  Text = auslesen("AT+RESET");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }
  delay(1000);  //So ein Reset dauert seine Zeit ...

  //Bricht alle Verbindungen ab und macht das Modul in der Bluetooth-Umgebung sichtbar
  Text = auslesen("AT+INIT");
  Serial.println(Text);
  Text = auslesen("AT+DISC");
  Serial.println(Text);

  //Gibt den Status des Moduls an
  Text = auslesen("AT+STATE?");
  Serial.println(Text);

  Text="";
  digitalWrite(LedPin, HIGH);
  Serial.println();
  Serial.println("Konfiguration erfolgreich!");
  Serial.println("Die gelbe LED an Pin13 sollte jetzt konstant leuchten.");
  Serial.println("Es können über die Konsole AT-Befehle an das Modul gesendet werden.");
  Serial.println("Der Bluetooth-Name des Moduls lautet: \"+Name+"\" , das Passwort lautet:\"1234\"");
  Serial.println("Das Modul kann jetzt auch mit einem Handy gekoppelt werden, alle empfangenen Texte werden angezeigt,
alle gesendeten Befehle gehen dann direkt ans Handy (wie im COM Modus).");
  Serial.println("Sollte der String \"LED an\" oder \"LED aus\" empfangen werden, reagiert die LED an Pin13 entsprechend
(egal ob vom seriellen Monitor, oder vom Handy).");
  Serial.println();

// -----  Für Temperaturmessung:  -----
  sensors.begin(); // Dallas Temperatur-Library für Nutzung der OneWire Library vorbereiten
}

void loop() {
  //Empfange Zeichen vom seriellen Monitor auf dem PC
 while(Serial.available() > 0){            // Solange etwas empfangen wird, durchlaufe die Schleife
    Zeichen = Serial.read();               // Speichere das empfangene Zeichen in der Variablen "Zeichen"
    Text.concat(Zeichen);                  // Füge das Zeichen an den String an, damit wir den kompletten Text erhalten
    if (Zeichen == "\n") {                 // War das letzte Zeichen ein NL (NewLine)?
      Serial.print(Text);                  // Schreibe den Text auf den seriellen Monitor
```

```
      HC05.print(Text);                    // Sende den empfangenen Text an das Modul (samt "\n")
                                           //Reagiere auf "LED an" bzw. "LED aus"
      if (Text.startsWith("Led an") || Text.startsWith("LED an") || Text.startsWith("LED AN")){
        digitalWrite(LedPin, HIGH);
      }
      if (Text.startsWith("Led aus") || Text.startsWith("LED aus") || Text.startsWith("LED AUS")){
        digitalWrite(LedPin, LOW);
      }
      Text="";                             // Lösche den String wieder für die nächste Nachricht
    }
 }
                                           //Empfange Zeichen vom HC-05 Modul
 while(HC05.available() > 0){              // Solange etwas empfangen wird, durchlaufe die Schleife
    Zeichen = HC05.read();                 // Speichere das empfangene Zeichen in der Variablen "Zeichen"
    Text.concat(Zeichen);                  // Füge das Zeichen an den String an, damit wir den kompletten Text erhalten
    if (Zeichen == "\n") {                 // War das letzte Zeichen ein NL (NewLine)?
      Serial.print(Text);                  // Sende den empfangenen Text an das Modul (samt "\n")
                                           //Reagiere auf "LED an" bzw. "LED aus"
      if (Text.startsWith("Led an") || Text.startsWith("LED an") || Text.startsWith("LED AN")){
        digitalWrite(LedPin, HIGH);
      }

      if (Text.startsWith("Led aus") || Text.startsWith("LED aus") || Text.startsWith("LED AUS")){
        digitalWrite(LedPin, LOW);
      }

      if (Text.startsWith("Temperatur") || Text.startsWith("temperatur") || Text.startsWith("temp")){
        sensors.requestTemperatures();                  // Temperatur abfragen
        delay(1000);
        Serial.print("Temperatur: ");
        Serial.print(sensors.getTempCByIndex(0) );      // Temperaturwert ausgeben
        Serial.println("°C");
        HC05.print("Temperatur: ");                     // Sende den empfangenen Text an das Modul
        HC05.print(sensors.getTempCByIndex(0) );        // Sende den empfangenen Text an das Modul
        HC05.print(" °C\n");                           // Sende den empfangenen Text an das Modul (samt "\n")
      }
      Text="";                                          // Lösche den String wieder für die nächste Nachricht
    }
 }
}


String auslesen(String Befehl){
  char Zeichen;                            // Jedes empfangene Zeichen kommt kurzzeitig in diese Variable
  String result="";
  Serial.println(Befehl);                  // Schreibe den übergebenen String auf den seriellen Monitor
  HC05.println(Befehl);                    // Sende den übergebenen String an das Modul
  delay(500);
  while(HC05.available() > 0){             // Solange etwas empfangen wird, durchlaufe die Schleife
    Zeichen = HC05.read();                 // Speichere das empfangene Zeichen in der Variablen "Zeichen"
    result.concat(Zeichen);                // Speichere die Antwort des Moduls
  }
  return result;                           // Übergebe die Rückantwort des Moduls
}
```

Bild 11 zeigt exemplarisch die Ausgaben im seriellen Monitor (Baudrate muss richtig eingestellt sein!), und Bild 12 zeigt die dazugehörige Ausgabe auf dem Smartphone.

In der hier verwendeten Android-App lassen sich bis zu 3 Reihen je 5 Buttons mit Makros belegen. Dadurch erhält man ohne Programmierung auf dem Smartphone sehr schöne Möglichkeiten für eine Oberfläche für Heimautomation (Lichter/Heizung schalten, Temperaturwerte oder Schalterstellungen abfragen usw.). Bild 13 gibt einen ersten Eindruck, wie die Makro-Buttons in der App dargestellt werden. Ein Tippen mit dem Finger auf einen Makro-Button sendet den hinter dem Makro hinterlegten Text per Bluetooth an das HC-05-Modul. Der obere Teil der App kann auch angepasst werden. Selbstverständlich kann man im Betrieb den PC weglassen (Bild 1 unten) und das Bluething Board beispielsweise in eine Lampe einbauen, um diese per Smartphone ein- und auszuschalten und die Temperatur abzufragen.

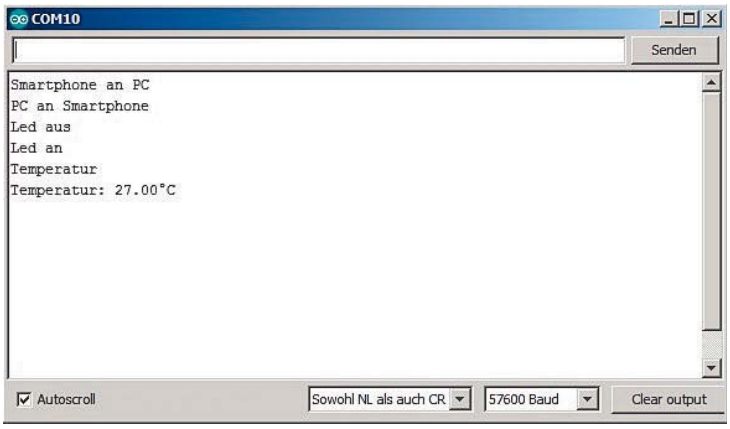


Bild 11: Kommunikation im seriellen Monitor

Bild 12: Terminal-App „Serial Bluetooth Terminal", Kommunikation

Bild 13: Terminal-App „Serial Bluetooth Terminal", Makros

Bild 14: Franzis' Lavalampen-App „Bluething"

Lavalampe

Ein schönes, ebenfalls mitgeliefertes Beispiel nutzt eine kostenlos aus dem Google Play Store ladbare App (Bild 14, [10]), um per Bluetooth eine RGB-Lampe zu steuern. Franzis nennt das Projekt „Lavalampe". Der prinzipielle Aufbau ist in Bild 15 zu sehen. Dort ist auf einem Steckbrett eine RGB-LED, also eine LED, die in einem Gehäuse eine rote, eine grüne und eine blaue LED eingebaut hat, über Vorwiderstände an die Pins 3, 5 und 6 des Arduino angeschlossen. Zuerst lädt man einen Sketch auf den Arduino, der den HC-05 initialisiert und dann lädt man den eigentlichen Lavalampen-Sketch auf den Arduino, der zum Betrieb benutzt wird.

Die Smartphone-App mit ihrer schönen Oberfläche (Bild 14) sendet permanent die Zahlen für die Farbanteile für Rot, Grün und Blau an das HC-05-Modul, das den Datenstrom seriell an den Arduino sendet. Der Arduino wertet den Datenstrom aus und schaltet die Pins für die LEDs entsprechend heller (Richtung 255) oder dunkler (Richtung 0). Der Datenstrom wird auch vom Arduino an den PC gesendet (Bild 16). Im Franzis-Installationspaket gibt es eine Anleitung für eine Lavalampe mit LED-Streifen. Statt die Lavalampe mit der Franzis-App zu steuern, könnte man auch eine Lampe in Abhängigkeit von Mess- oder Temperaturwerten in unterschiedlichen Farben leuchten lassen.

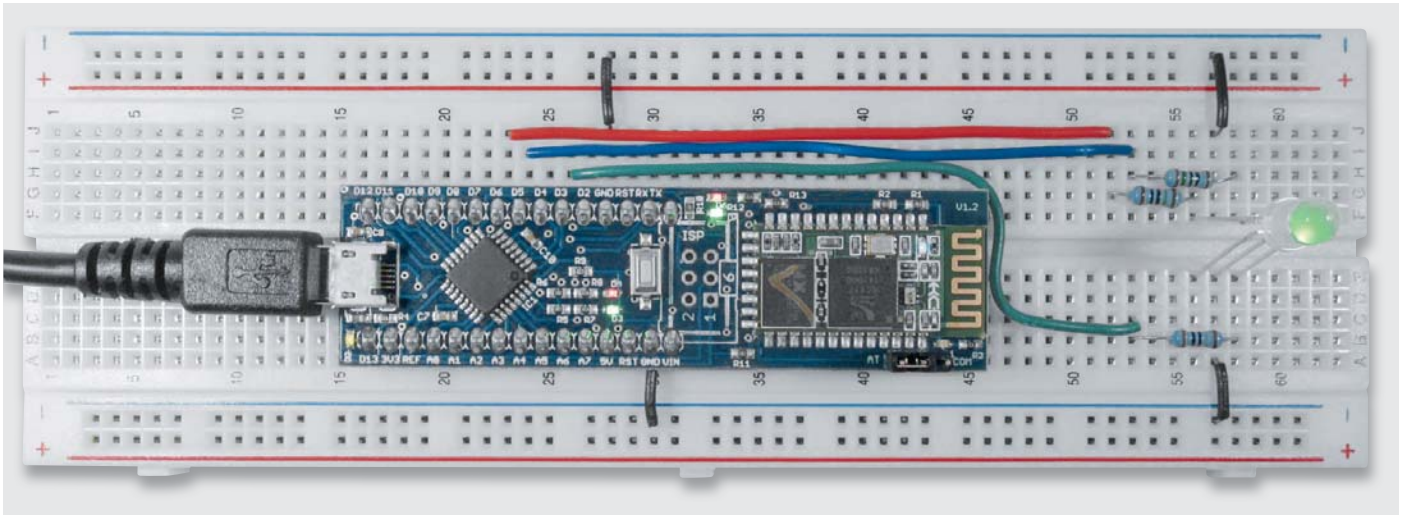
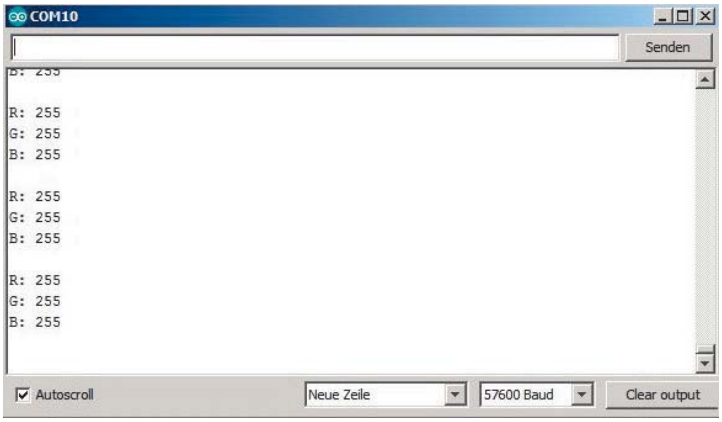


Bild 15: Lavalampe auf Steckbrett

Bild 16: Datenstrom Lavalampe



Lavalampe, Initialisierung

```
/*
 * Lavalampe - Initialisierung
 * Der Jumper muss auf AT stehen
 */

// Variablen zuweisen
String Text="";    // In diesem String werden Texte zwischengespeichert
char Zeichen;      // In dieser Variable werden Zeichen zwischengespeichert
#define rxPin 11   // Virtueller RX-Pin des Moduls zur Kommunikation mit dem HC-05
#define txPin 12   // Virtueller TX-Pin des Moduls zur Kommunikation mit dem HC-05

#include <SoftwareSerial.h>
SoftwareSerial HC05(txPin, rxPin);

void setup() {
  Serial.begin(57600);     // Definieren der Baudrate für die integrierte serielle Schnittstelle
  HC05.begin(38400);       // Definieren der Baudrate für die Software-simulierte serielle Schnittstelle

  initialisiere();
}

void loop() {
  // put your main code here, to run repeatedly:

}

void initialisiere(){
  Serial.println("Initialisiere...");
  Serial.println();
  Serial.println("Konfiguriere HC-05 Modul:");
  Serial.println();
    //Reset HC-05 -> Slave mode, Baudrate 38400, Passwort:1234, Device-Name: "hc01.com HC-05"
  Text = auslesen("AT+ORGL");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    Serial.println("Steht der Jumper auf AT?");
    Serial.println("Sind alle Bluetooth-Verbindungen zum Modul getrennt?");
    Serial.println("Wechsle in den COM-Modus...");
    Serial.println();
    return;
  }

  //Alle Devices aus der Pair-Liste löschen
  Text = auslesen("AT+RMAAD");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }
```

```
  //Name des Moduls auf "Lavalampe" setzen
  Text = auslesen("AT+NAME=Lavalampe");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }


  //Modul neustarten und eventuelle Verbindungen resetten
  Text = auslesen("AT+RESET");
  Serial.println(Text);
  if (!Text.startsWith("OK")) {
    Serial.println("Fehler!");
    while(1);
  }
  delay(1000);  //Das Reset benötigt ein wenig Zeit

  //Bricht alle Verbindungen ab und macht das Modul in der Bluetooth-Umgebung sichtbar
  Text = auslesen("AT+INIT");
  Serial.println(Text);
  Text = auslesen("AT+DISC");
  Serial.println(Text);

 Serial.println("Das Modul ist jetzt bereit, bitte den Jumper auf COM stecken und den Sketch Lavalampe.ino auf den Chip uebertragen");
}

String auslesen(String Befehl){
  char Zeichen;                          // Jedes empfangene Zeichen kommt kurzzeitig in diese Variable
  String result="";
  Serial.println(Befehl);                // Schreibe den übergebenen String auf den seriellen Monitor
  HC05.println(Befehl);                  // Sende den übergebenen String an das Modul
  delay(500);
  while(HC05.available() > 0){           // Solange etwas empfangen wird, durchlaufe diese Schleife
    Zeichen = HC05.read();               // Speichere das empfangene Zeichen in der Variablen "Zeichen"
    result.concat(Zeichen);              // Speichere die Antwort des Moduls
  }
  return result;                         // Übergebe die Rückantwort des Moduls
}
```

Lavalampe, Runtime

```
/*
 * Lavalampe
 * Die Buchstaben (R,G,B) stehen für die Farben Rot, Grün, Blau
 * Die Zahlen (0-255) geben den Helligkeitswert wieder
 * Die Raute dient als Terminator
 */

// Variablen zuweisen
char Zeichen;      // Jedes empfangene Zeichen kommt kurzzeitig in diese Variable
String Text;       // In diesem String speichern wir dann unseren kompletten Text
#define rxPin 11   // Virtueller RX-Pin des Moduls zur Kommunikation mit dem HC-05
#define txPin 12   // Virtueller TX-Pin des Moduls zur Kommunikation mit dem HC-05
#define gruen 3    // Die Farbe Grün wird dem PIN 3 zugewiesen
#define blau 5     // Die Farbe Blau wird dem PIN 5 zugewiesen
#define rot 6      // Die Farbe Rot wird dem PIN 6 zugewiesen


#include <SoftwareSerial.h>
SoftwareSerial HC05(txPin, rxPin);

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  HC05.begin(38400);      // Definieren der Baudrate für die Software-simulierte serielle Schnittstelle

  pinMode(gruen, OUTPUT);
  pinMode(rot, OUTPUT);
  pinMode(blau, OUTPUT);
```

```
  //LED-Test:
  analogWrite(rot, 255);
  analogWrite(gruen, 0);
  analogWrite(blau, 0);
  delay(1000);
  analogWrite(rot, 0);
  analogWrite(gruen, 255);
  analogWrite(blau, 0);
  delay(1000);
  analogWrite(rot, 0);
  analogWrite(gruen, 0);
  analogWrite(blau, 255);
  delay(1000);
  analogWrite(rot, 255);
  analogWrite(gruen, 255);
  analogWrite(blau, 255);
}

void loop() {
  while(HC05.available() > 0){                // Solange etwas empfangen wird, durchlaufe die Schleife
    Zeichen = HC05.read();                    // Speichere das empfangene Zeichen in der Variablen
                                                 "Zeichen"
    if (Zeichen!="#") Text.concat(Zeichen);   // Füge das Zeichen an den String an, damit wir den
                                                 kompletten Text erhalten
    if (Zeichen == "#") {                     // War das letzte Zeichen eine Raute?
      if (Text.startsWith("R")) {
        Text.replace("R", "");
        Serial.print("R: ");
        Serial.println(Text.toInt());
        analogWrite(rot, Text.toInt());
      }
      if (Text.startsWith("G")) {
        Text.replace("G", "");
        Serial.print("G: ");
        Serial.println(Text.toInt());
        analogWrite(gruen, Text.toInt());
      }
      if (Text.startsWith("B")) {
        Text.replace("B", "");
        Serial.print("B: ");
        Serial.println(Text.toInt());
        Serial.println();
        analogWrite(blau, Text.toInt());
      }
      Text="";                                // Lösche den String für die nächste Nachricht
    }
  }
}
```

Fazit

Mit dem Bluething Board von Franzis lassen sich durch den eingebauten Arduino Nano sehr schnell mit keinen oder geringen Vorkentnissen Bluetooth-Funkverbindungen für eigene Projekte aufbauen. Kommunikationspartner des Boards kann ein Android Smartphone, ein PC oder ein weiteres Bluething-Modul sein. Durch die sehr guten (per Download) mitgelieferten Beispiele ist der Einstieg in die Benutzung des Bluetooth-Moduls sehr einfach. Im vorliegenden Artikel wurde das Modul im sogenannten Slave-Modus verwendet und ein Smartphone als Kommunikationspartner verwendet. Ebenso kann ein PC als Kommunikationspartner benutzt werden. Im Lieferumfang (Download) ist auch ein Beispiel für die Kommunikation von einem Bluething Board zu einem anderen Bluething Board. Eines der Module wird dabei als Slave benutzt und eines als Master. Durch viele freie I/O-Pins lassen sich Sensoren und Aktoren leicht an das Board anschließen. Neben der Anwendung im Heimautomationsbereich können auch beispielsweise Messwerte von einer Messanordnung zu einem PC, Tablet oder Smartphone übertragen werden. ELV

Weitere Infos:

[1] Schaltplan Arduino Nano:
https://www.arduino.cc/en/uploads/Main/ArduinoNano30Schematic.pdf

[2] HC-05 Datenblatt:
ftp://imall.iteadstudio.com/Modules/IM120723009/DS_IM120723009.pdf

[3] Bluetooth:
https://www.bluetooth.com sowie https://de.wikipedia.org/wiki/Bluetooth

[4] Franzis Bluething Board:
http://www.franzis.de/maker/raspberry-pi-arduino-und-mehr/internet-of-things-bluething-board-platine

[5] Kurzanleitung Bluething Board bei ELV:
https://www.elv.de: Webcode #10143

[6] Arduino-Einführung:
https://www.elv.de: Webcode #10144

[7] Arduino-Software:
https://www.arduino.cc/en/Main/Software#

[8] Terminalprogramm HTerm:
http://www.der-hammer.info/terminal/

[9] Android Terminal App:
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

[10] Franzis Bluething App:
https://play.google.com/store/apps/details?id=appinventor.ai_tobias_stuckenberg.Bluething&hl=de

Preisstellung Oktober 2017 – aktuelle Preise im ELV Shop

Empfohlene Produkte	Best.-Nr.	Preis
USB-Modul UM2102N, Komplettbausatz	CQ-15 09 52	€ 5,95
Franzis Bluething Board	CQ-14 48 59	€ 49,95
Vellemann Bluetooth-Modul HC-05	CQ-13 32 97	€ 14,95
Franzis Arduino Nano Board	CQ-14 48 60	€ 19,95
Franzis Arduino Uno-Platine R3	CQ-10 29 70	€ 27,95
Franzis Arduino Leonardo-Platine	CQ-10 87 20	€ 24,95
Franzis Arduino Pretzel Board IoT WiFi Board	CQ-12 23 18	€ 29,95
USB-2.0-Hi-Speed-Kabel, schwarz, 0,6 m	CQ-12 00 51	€ 3,25
Steckplatine 801, 400 Kontakte	CQ-12 59 05	€ 4,95
Steckplatine 102, 830 Kontakte	CQ-12 59 03	€ 6,95
LED LF5WAEMBGMBW, 5 mm, RGB, 20–30 mcd	CQ-09 96 47	€ 2,–
Wasserdichter Sensor mit 2,10 m Zuleitung für TS 125/TSM 125	CQ-10 27 83	€ 12,95
Franzis Arduino Lernpaket + Arduino UNO-Platine	CQ-12 46 92	€ 79,95
Vellemann Steckbrücken-Set, Stecker auf Buchse, 15 cm, 10-teilig	CQ-12 90 19	€ 2,50
Vellemann Steckbrücken-Set, Buchse auf Buchse, 15 cm, 10-teilig	CQ-12 90 20	€ 2,50
Vellemann Steckbrücken-Set, Stecker auf Stecker, 15 cm, 10-teilig	CQ-12 90 18	€ 2,50
ELV Kabelsatz für Steckplatinen, 140-teilig	CQ-12 59 06	€ 3,95
AREXX Schaltdraht-Sortiment (10 Drähte): 0,5 mm², 60 m	CQ-05 47 68	€ 9,95
Knipex Super-Knips Electronic Form 1 mit Drahtklemme	CQ-08 17 12	€ 17,95
AREXX Schaltlitze-Sortiment (10 Litzen): 0,5 mm², 60 m	CQ-05 47 96	€ 8,95