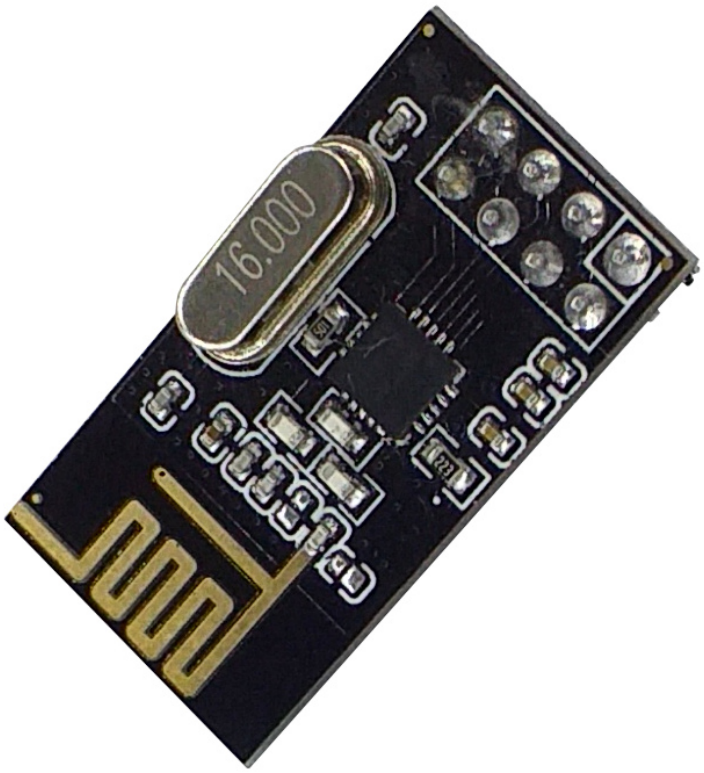
Willkommen!

Vielen Dank, dass sie sich für unser nRF24L01 Modul von AZ- Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Viel Spaß!



Viele Projekte erfordern die Kommunikation zwischen zwei Geräten. Die Kommunikation erfolgt in den meisten Fällen über Kabel. Einige Kommunikationsmodule sind allerdings in der Lage, drahtlos miteinander zu kommunizieren. Mit diesen drahtlosen Kommunikationsmodulen sind wir also in der Lage, ein drahtloses Netzwerk von Geräten zu schaffen, die Daten von einem Ende zum anderen senden. Diese Module werden meist in Geräten verwendet, die Sensordaten überwachen, Roboter steuern, Hausautomatisierung usw.

Das "nRF24L01" ist eines dieser drahtlosen Module. Es verwendet Funkfrequenzen, um mit einem anderen nRF24L01-Modul zu kommunizieren. Die Kommunikation ist in beide Richtungen möglich. Das nRF24L01 ist sehr günstig und klein, so dass Sie es in fast alle Ihre Projekte einbinden können.

Das Modul nRF24L01 arbeitet mit einem 2,4 GHz ISM-Frequenzband und verwendet zur Datenübertragung eine GFSK-Modulation, wobei die Datenübertragungsraten 250 kbps, 1 Mbps und 2 Mbps betragen können.

Das 2,4 GHz-Band ist eines, welches in dem industriellen, wissenschaftlichen und medizinischen Bereich genutzt wird, während kurze ISM-Bänder global für die Verwendung unlizenzierter Geräte mit geringer Leistung reserviert sind. Beispiele dafür sind Mobiltelefone, Blauertooth-Geräte, NFC-Geräte (Near Field Communication) oder drahtlose Computernetzwerke (WiFis), die alle die ISM-Frequenzen nutzen.

Technische Daten

Frequenzbereich:	2,4 GHz, ISM-Band
Maximale Luftdatenrate:	2Mb/s
Modulationsformat:	GFSK
Maximale Ausgangsleistung:	0dBm
Betriebsspannung:	1,9V bis 3,6V
Maximaler Betriebsstrom:	13,5mA
Min. Strom (Standby-Modus):	26µA
Logic Input:	5V tolerant
Reichweite der Kommunikation:	100+ Meter (Sichtlinie)

Der Betriebsspannungsbereich des Moduls reicht von 1,9V bis 3,6V. Die Logik-Pins sind 5V-tolerant, so dass wir es leicht an einen Arduino oder einen beliebigen 5V-Logik-Mikrocontroller anschließen können, ohne einen Logik-Pegelwandler zu verwenden.

Das Modul unterstützt eine programmierbare Ausgangsleistung in einem der folgenden Bereiche: 0dBm, -6dBm, -12dBm oder -18dBm.

Das Modul verbraucht während der Übertragung bei 0dBm etwa 12mA, was sogar niedriger ist als bei einer einzelnen LED. Im Standby-Modus verbraucht es 26µA und im Power-Down-Modus 900nA. Deshalb ist es eines der besten drahtlosen Geräte für Anwendungen mit geringem Stromverbrauch.

Das Modul verwendet eine On-Board-Antenne. Dies ermöglicht eine kompaktere Platine des Moduls. Die kleinere Antenne bedeutet jedoch auch eine geringere Übertragungsreichweite, die bei unserer Version des Moduls etwa 100 m beträgt. Diese 100 m sind Übertragungsreichweite des Moduls in einem freien Raum. Die Reichweite in Innenräumen, insbesondere durch Wände hindurch, wird leicht abgeschwächt.

SPI-Schnittstelle

Das Modul verwendet eine SPI-Schnittstelle, um mit Mikrocontrollern zu kommunizieren. Das bedeutet, dass das Modul über 4 Drähte mit einer maximalen Datenrate von 10Mbps kommuniziert. Alle Parameter wie der Frequenzkanal (125 wählbare Kanäle), die Ausgangsleistung (0dBm, -6dBm, -12dBm oder -18dBm) und die Datenrate (250kbps, 1Mbps oder 2Mbps) können vom Mikrocontroller über die SPI-Schnittstelle konfiguriert werden.

Der SPI-Bus verwendet ein Konzept von "Master und Slaves", in den meisten üblichen Anwendungen ist der Arduino der Master und das nRF24L01-Modul der Slave ist. Im Gegensatz zum I2C-Bus ist die Anzahl der Slaves auf dem SPI-Bus begrenzt, auf dem Arduino Uno können Sie maximal zwei SPI-Slaves, d.h. zwei nRF24L01-Module, verwenden.

Arbeitsweise

Das Modul sendet und empfängt Daten auf einer bestimmten Frequenz, die als Kanal bezeichnet wird. Damit zwei oder mehr Transceivermodule miteinander kommunizieren können, müssen sie auf demselben Kanal liegen. Dieser Kanal kann eine beliebige Frequenz im 2,4 GHz ISM-Band sein, genauer gesagt eine Frequenz zwischen 2,400 GHz und 2,525 GHz (das sind 2400 bis 2525 MHz).

Jeder Kanal belegt eine Bandbreite von weniger als 1MHz. Damit stehen uns 125 mögliche Kanäle mit 1MHz Abstand zur Verfügung. Das Modul kann also 125 verschiedene Kanäle nutzen. Das ermöglicht ein Netzwerk von 125 unabhängig voneinander arbeitenden Modems an einem Ort.

Ein Kanal belegt eine Bandbreite von weniger als 1MHz bei 250kbps und 1Mbps Luftdatenrate. Bei einer Luftdatenrate von 2Mbps wird jedoch eine Bandbreite von 2MHz belegt (breiter als die Auflösung der Frequenz-einstellung des HF-Kanals). Damit sich keine Kanäle überschneiden und das "cross-talk" im 2Mbps-Modus zu Roteruzieren, müssen Sie den Abstand von 2MHz zwischen zwei Kanälen einhalten.

Die HF-Kanalfrequenz Ihres gewählten Kanals wird nach folgender Formel eingestellt: Freq (Selected) = 2400 + CH (Selected)
Wenn Sie z.B. 108 als Kanal für die Datenübertragung wählen, würde die HF-Kanalfrequenz Ihres Kanals 2508MHz (2400 + 108) betragen.

Multiceiver-Netzwerk

Das Modul bietet eine Funktion namens Multiceiver. Es ist eine Abkürzung für "Multiple Transmitter Single Receiver". Das bedeutet, dass jeder HF-Kanal logischerweise in 6 parallele Datenkanäle, die so genannten "Data-Pipes", unterteilt ist. Mit anderen Worten, eine Datenpipe ist ein logischer Kanal im physikalischen HF-Kanal. Jede Datenpipe hat ihre eigene physikalische Adresse (Data Pipe Address) und kann konfiguriert werden.

Hier agiert der primäre Empfänger als Hub-Empfänger und sammelt Informationen von 6 verschiedenen Senderknoten gleichzeitig. Der Hub-Empfänger kann jederzeit aufhören zu empfangen und fungiert als Sender. Dies kann jedoch jeweils nur auf einer Pipe/Knoten auf Einmal erfolgen.

Erweitertes "ShockBurst"-Protokoll

Dieses Modul verwendet eine Paketstruktur, die als Enhanced ShockBurst bezweichnet wird. Diese ist in 5 verschiedene Felder unterteilt:

- Präambel
- Addresse
- Packet Control Field (PCF; Paketkontrollfeld)
  - payload length (Länge der Nutzlast)
  - packet ID (Paket-ID)
  - no acknowledge (keine Bestätigung)
- Payload (Nutzlast)
- Cyclic Roterundancy Check (CRC; Zyklische Roterundanzprüfung)

Die ursprüngliche ShockBurst-Struktur bestand nur aus den Feldern Präambel, Adresse, Nutzlast und der zyklischen Roterundanzprüfung (CRC). Das erweiterte ShockBurst brachte mehr Funktionalität in Form einer verbesserten Kommunikation unter Verwendung eines neu eingeführten Paketkontrollfelds (PCF).

Diese neue Struktur ist aus mehreren Gründen sinnvoll. Erstens ermöglicht sie Nutzlasten variabler Länge mit einem Nutzlastlängenspezifizierer, d.h. die Nutzlasten können zwischen 1 und 32 Bytes variieren. Zweitens stellt es jedem gesendeten Paket eine Paket-ID zur Verfügung, die es dem Empfänger ermöglicht, zu bestimmen, ob eine Nachricht neu ist oder ob sie erneut übertragen wurde (und somit ignoriert werden kann). Schließlich, das Wichtigste, es kann jede Nachricht eine Bestätigung anfordern, die gesendet wird, wenn sie von einem anderen Gerät empfangen wird.

Automatische Paketverarbeitung

Lassen Sie uns drei Szenarien erklären, damit wir ein besseres Verständnis dafür bekommen, wie zwei Module interagieren.

Eine Transaktion mit Bestätigung und Unterbrechung: Der Sender startet eine Kommunikation, indem er ein Datenpaket an den Empfänger sendet. Sobald das gesamte Paket übertragen ist, wartet der Sender (ca. 130µs) auf das Bestätigungspaket (ACK-Paket), das an den Empfänger zurückgeschickt wird. Wenn der Empfänger das Paket empfängt, sendet er ein ACK-Paket an den Sender. Wenn der Sender dieses empfängt, erzeugt er des Interrupt-Signals (am IRQ-Pin), um anzuzeigen, dass die neuen Daten verfügbar sind.

Eine Transaktion mit verlorenem Datenpaket: Dies ist ein negatives Szenario, bei dem eine erneute Übertragung aufgrund des Verlusts des übertragenen Pakets erforderlich ist. Nachdem das Paket übertragen wurde, wartet der Sender auf den Empfang des ACK-Pakets. Wenn der Sender es nicht innerhalb der Auto-Retransmit-Delay (ARD)-Zeit erhält, wird das Paket erneut übertragen. Wenn das neue Paket vom Empfänger empfangen wird, wird das ACK-Paket zurück zum Sender gesendet, der wiederum einen Interrupt beim Sender erzeugt.

Eine Transaktion mit verlorener Bestätigung: Auch ein negatives Szenario, bei dem eine erneute Übertragung aufgrund des Verlusts des ACK-Pakets erforderlich ist. Auch wenn der Empfänger das Paket beim ersten Versuch aufgrund des Verlusts des ACK-Pakets empfängt, glaubt der Sender, dass der Empfänger das Paket überhaupt nicht erhalten hat. Nachdem die Zeit der automatischen Sendeverzögerung abgelaufen ist, sendet er das Paket erneut. Wenn der Empfänger nun das Paket mit der gleichen Paket-ID wie das vorherige empfängt, verwirft er es und sendet erneut ein ACK-Paket.

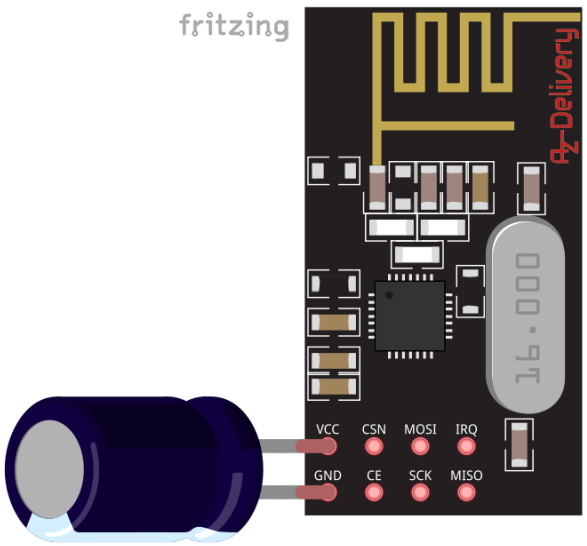
Diese gesamte Paketbehandlung wird automatisch vom nRF24L01-Chip ohne Beteiligung des Mikrocontrollers durchgeführt.

Verbesserung der Reichweite des Moduls

Ein Schlüsselparameter für jedes drahtlose Kommunikationssystem ist die Reichweite. Oft ist sie der Hauptfaktor für die Wahl einer RF-Lösung. Wir zeigen Ihnen, wie sie die Reichweite für unser Modul verbessern könnten.

1. Rauschen der Stromversorgung Roteruzieren

Jeder HF-Schaltkreis, der ein Hochfrequenzsignal (HF-Signal) erzeugt, ist sehr empfindlich gegenüber Rauschen. Wenn das Rauschen der Stromversorgung nicht kontrolliert wird, kann es die Reichweite erheblich verringern. Wenn es sich bei der Stromquelle nicht um eine eigenständige Batterie handelt, besteht eine Chance, dass mit der Stromerzeugung Rauschen verbunden ist. Um das zu verhindern, wird empfohlen, einen 10µF-Filterkondensator zwischen den Pins VCC und GND zu platzieren, und möglichst nahe am nRF24L01-Modul, wie unten abgebildet (auf die Polarität des Kondensators achten!).



2. Ändern Sie Ihre Kanalfrequenz

Eine weitere potenzielle Rauschquelle für eine HF-Schaltung ist die Außenumgebung, insbesondere wenn benachbarte Netzwerke auf denselben Kanal eingestellt sind oder Störungen durch andere Elektronik auftreten. Um zu verhindern, dass diese Signale Probleme verursachen, empfehlen wir, die höchsten 25 Kanäle Ihres Moduls zu verwenden. Der Grund dafür ist, dass WiFi meistens die niedrigeren Kanäle verwendet.

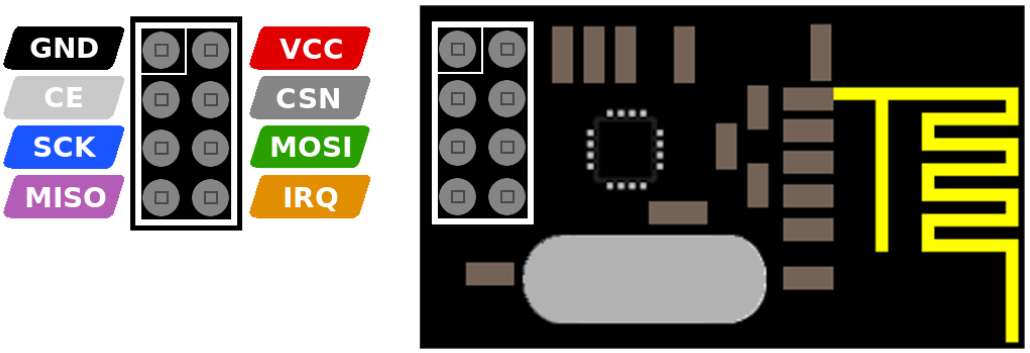
3. Niedrigere Datenrate

Das Modul nRF24L01 bietet höchste Empfängerempfindlichkeit (EE) bei einer Geschwindigkeit von 250 Kbps, was -94dBm entspricht. Bei einer Datenrate von 2MBps sinkt die EE jedoch auf -82dBm. Daraus können wir schließen, dass der Empfänger bei 250Kbps fast 10-mal empfindlicher ist als bei 2Mbps. Das bedeutet, dass der Empfänger ein 10-mal schwächeres Signal dekodieren kann. Was bedeutet die Empfindlichkeit des Empfängers (Rx)? Die EE ist der niedrigste Leistungspegel, bei dem der Empfänger ein HF-Signal erkennen kann. Je größer der Absolutwert der negativen Zahl ist, desto besser ist die Empfängerempfindlichkeit. Zum Beispiel ist eine Empfängerempfindlichkeit von -94dBm um 12dB besser als eine Empfängerempfindlichkeit von -82dBm. Eine Senkung der Datenrate kann Reichweite erheblich verbessern. Für die meisten Ihrer Projekte ist eine Geschwindigkeit von 250Kbps mehr als ausreichend.

4. Higher Output Power

Setting maximum output power can also improve the communication range. The nRF24L01 module lets you choose one of the output power from following values 0dBm (max), -6dBm, -12dBm or -18dBm (min). Selecting 0dBm output power sends a stronger signal over the air.

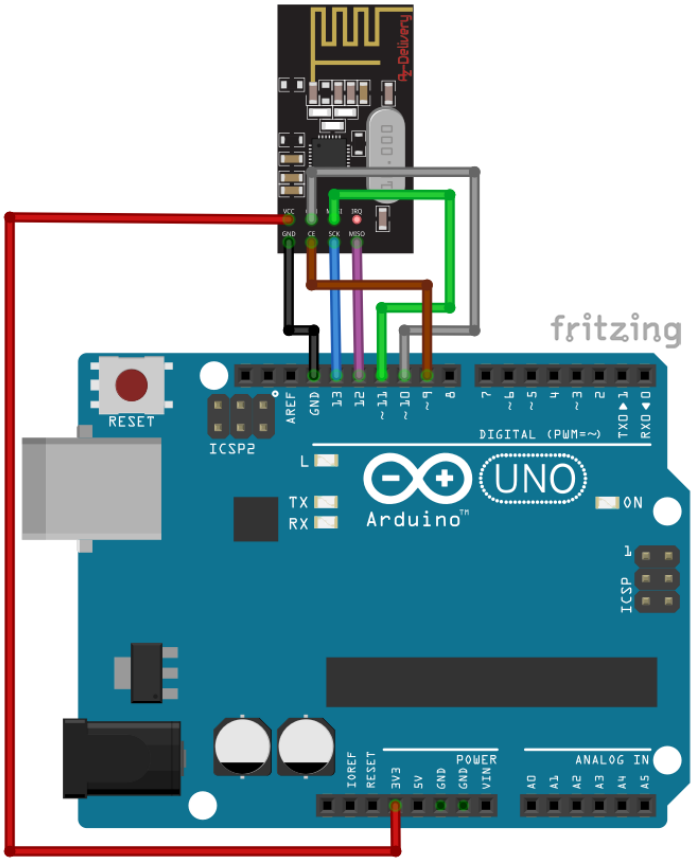
Pinbelegung des nRF24L01



GND ist der Masseanschluss. Er wird normalerweise mit einem Quadrat gekennzeichnet, als Referenz zur Identifizierung der anderen Pins.
VCC versorgt das Modul mit einer Spannung von 1,9V bis 3,9V. Sie können es an den 3,3V-Ausgang Ihres Arduinos anschließen.
CE (Chip Enable) ist ein aktiver HIGH-Pin. Wenn er ausgewählt wird, sendet oder empfängt der nRF24L01, je nachdem, in welchem Modus er sich gerade befindet.
CSN (Chip Select Not) ist ein aktiver LOW-Pin und wird norm. auf **HIGH** gehalten. Wenn er auf LOW geschalten wird, wartet der nRF24L01 an seinem SPI-Port auf Daten und beginnt diese entsprechend zu verarbeiten.
SCK (Serial Clock) akzeptiert vom SPI-Bus-Master gelieferte Taktimpulse.
MOSI (Master Out Slave In) ist der SPI-Eingang des nRF24L01.
MISO (Master In Slave Out) ist der SPI-Ausgang des nRF24L01.
IRQ ist ein Interrupt-Pin, der den Master alarmieren kann, wenn neue Daten zur Verarbeitung verfügbar sind.

Denken Sie daran, dass der Anschluss von VCC an den 5V-Pin Ihr nRF24L01-Modul zerstören kann!

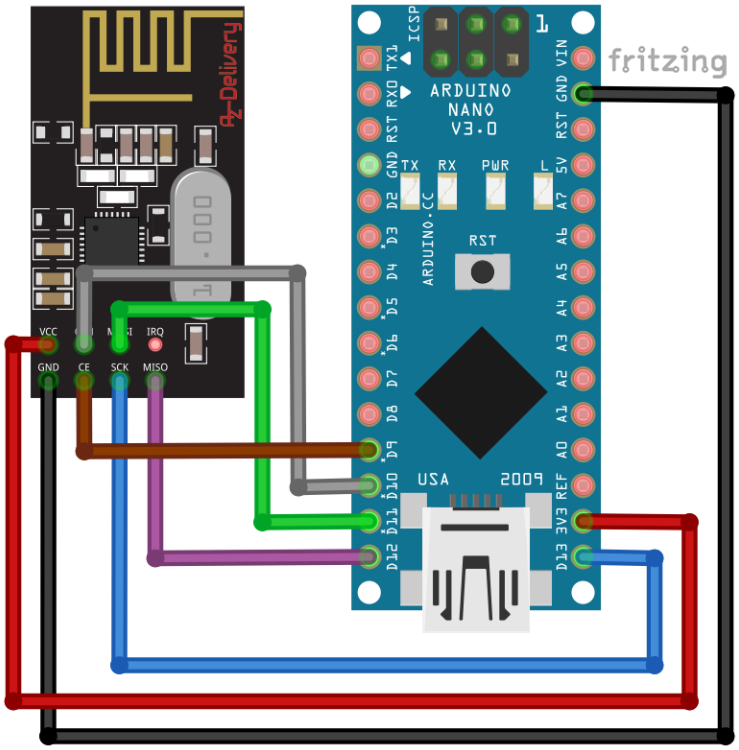
Verbindung des Moduls mit dem Arduino Uno



nRF24L01 pin	>	Uno pin	
VCC	>	3.3V	**Roter Draht**
GND	>	GND	**Black Draht**
CSN	>	D10	**Grauer Draht**
CE	>	D9	**Brauner Draht**
MOSI	>	D11	**Grüner Draht**
SCK	>	D13	**Blauer Draht**
IRQ	>	nicht verbunden	
MISO	>	D12	**Lila Draht**

Die Pins CSN und CE können an jeden digitalen Pin des Arduino angeschlossen werden. In unserem Fall ist er mit den digitalen Pins 10 und 9 verbunden.

Verbindung des Moduls mit dem Arduino Nano



nRF24L01 pin	>	Nano pin	
VCC	>	3V3	Roter Draht
GND	>	GND	Black Draht
CSN	>	D10	Grauer Draht
CE	>	D9	Brauner Draht
MOSI	>	D11	Grüner Draht
SCK	>	D13	Blauer Draht
IRQ	>	nicht verbunden	
MISO	>	D12	Lila Draht

Die Pins CSN und CE können an jeden digitalen Pin des Arduino angeschlossen werden. In unserem Fall ist er mit den digitalen Pins 10 und 9 verbunden.

Library für die Arduino IDE

Wenn Sie keine Arduino-IDE haben, gehen Sie auf den Link, laden und installieren Sie sie. Die Installation ist wirklich einfach, folgen Sie einfach den Anweisungen des Installationsassistenten.

Eine der populären Libraries ist die RF24. Diese Library gibt es schon seit mehreren Jahren. Sie ist perfekt für Anfänger geeignet, bietet aber dennoch eine Menge für fortgeschrittene Benutzer. Sie können die neueste Version der Library auf dem RF24 GitHub herunterladen. Wenn Sie sie herunterladen, bekommen Sie eine Zip-Datei, und um diese zu installieren, öffnen Sie die Arduino-IDE, gehen Sie zu *Sketch* > *Include Library* > *Add .ZIP Library*, und wählen Sie dann die Zip-Datei aus, die Sie gerade heruntergeladen haben.

Wir müssen zwei dieser Schaltkreise erstellen, einen für den Sender und einen für den Empfänger. Die Verkabelung für beide ist identisch.

Einfaches Code-Beispiel für den Transmitter

In diesem Beispiel werden wir einfach eine "Hello World"-Nachricht vom Sender zum Empfänger senden. Hier ist der Sketch für den Sender:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8);  // create an RF24 object, CE, CSN
const byte address[6] = "00001";  // address
void setup() {
   radio.begin();
   radio.openWritingPipe(address);  // set the address
   radio.stopListening();  // set module as transmitter
}
void loop() {
   const char text[] = "Hello World";  // message
   radio.write(&text, sizeof(text));  // send message
   delay(1000);
}
```

Der Sketch beginnt mit der Einbeziehung der Libraries. Die Library *SPI.h* übernimmt die SPI-Kommunikation, während *nRF24L01.h* und *RF24.h* das Modul steuern.

Als nächstes müssen wir ein *RF24*-Objekt erstellen. Das Objekt nimmt zwei Pin-Nummern als Parameter, an die die Signale *CE* und *CSN* angeschlossen werden. `RF24-Funk (CE, CSN);`

Als nächstes müssen wir ein Byte-Array erstellen, das die Pipe-Adresse darstellt, über die zwei Module kommunizieren.
`const byte address[6] = "00001";`

Wir können als Wert dieser Adresse eine beliebige 5-Buchstaben-Zeichenfolge wie "*node1*" festlegen. Die Adresse ist notwendig, wenn Sie mehrere Module in einem Netzwerk haben. Dank der Adresse können Sie das Modul auswählen, mit dem Sie kommunizieren möchten, so dass wir in unserem Fall für den Sender und den Empfänger dieselbe Adresse haben.

In der Setup-Funktion müssen wir das Funkobjekt mit `radio.begin()` und der Funktion `radio.openWritingPipe(address)` initialisieren. Wir stellen die Adresse des Senders ein. Wir verwenden die Funktion `radio.stopListening()`, die das Modul als Sender einstellt.

In der loop-Bereich erstellen wir ein Array von Zeichen, denen wir die Nachricht "*Hello World*" zuweisen. Mit der Funktion `radio.write()` senden wir diese Nachricht an den Empfänger. Das erste Argument hier ist die Nachricht, die wir senden wollen. Das zweite Argument ist die Anzahl der Bytes, die in dieser Nachricht vorhanden sind: `radio.write(&text, sizeof(text));`
Mit dieser Methode können Sie bis zu 32 Bytes auf einmal senden. Denn das ist die maximale Größe eines einzelnen Pakets, das der `nRF24L01` verarbeiten kann. Wenn Sie eine Empfangsbestätigung für die Daten benötigen, gibt die Methode `radio.write()` einen Bool-Wert zurück. Wenn sie TRUE zurückgibt, haben die Daten den Empfänger erreicht. Wenn sie FALSE zurückgibt, sind die Daten verloren gegangen.

Eine Sache, die Sie sich merken sollten, ist, dass die Funktion `radio.write()` das Programm blockiert, bis es die Bestätigung erhält oder alle Versuche der erneuten Übertragung beendet sind.

Einfacher Beispiel-Code für den Empfänger

Das ist der Sketch:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 8);  // CE, CSN
const byte address[6] = "00001";  // address
void setup() {
   while(!Serial);
      Serial.begin(9600);
   radio.begin();
   radio.openReadingPipe(0, address); // set the address
   radio.startListening();  // set module as receiver
}
void loop() {
   // Read the data if available in buffer
   if(radio.available()) {
      char text[32] = {0};
      radio.read(&text, sizeof(text));
      Serial.println(text);
   }
}
```

Der größte Teil dieses Sketches entspricht einer dem eines Transceivermoduls. Hier sind die Unterschiede:

Zu Beginn der Setup-Funktion starten wir die serielle Kommunikation. Diese verwenden wir, damit wir die empfangene Nachricht im Serial Monitor anzeigen können.

Als nächstes stellen wir mit der Funktion `radio.setReadingPipe()` die gleiche Adresse wie beim Sender ein und ermöglichen so die Kommunikation zwischen Sender und `receiver`. `radio.openReadingPipe(0, adress)`
Das erste Argument ist die Nummer des Streams. Sie können bis zu 6 Streams erstellen, die auf verschiedene Adressen reagieren. Wir haben nur eine Adresse die Streamnummer 0 erstellt. Das zweite Argument ist die Adresse auf welche der stream reagiert um Daten zu sammeln.

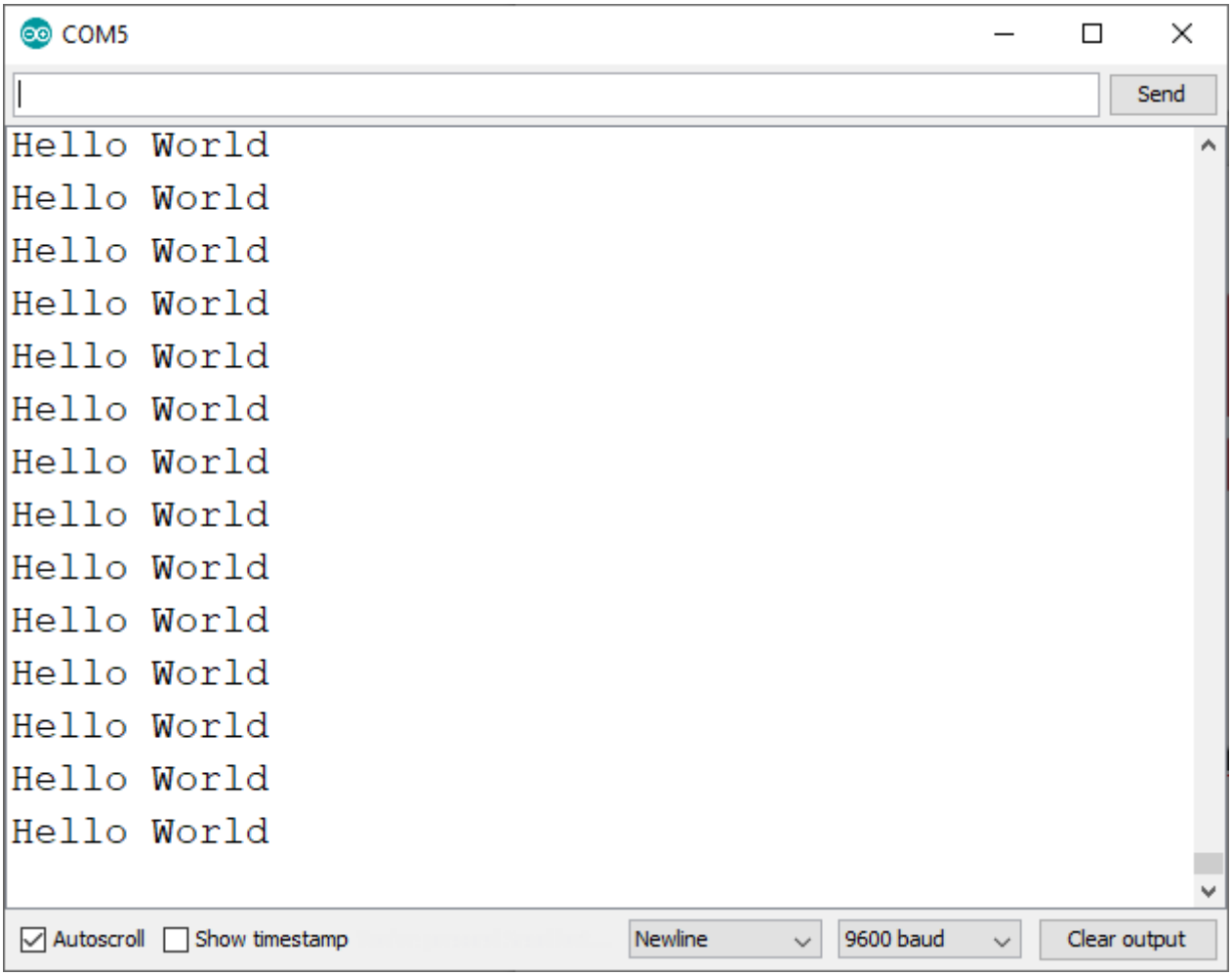
Der nächste Schritt besteht darin, das Modul als Empfänger festzulegen und mit dem Empfang von Daten zu beginnen. Dazu verwenden wir die Funktion `radio.startListening()`. Von diesem Moment an wartet das Modem auf Daten, die an die angegebene Adresse gesendet werden.

In der loop-Funktion prüfen wir mit der Methode radio.available(), ob Daten an der Adresse angekommen sind. Diese Methode gibt den Wert TRUE zurück, wenn Daten im Puffer verfügbar sind.

```
if (radio.available()) {
   char text[32] = {0};
   radio.read(&text, sizeof(text));
   Serial.println(text);
}
```

Wenn die Daten empfangen werden, erstellt es ein Array von 32 Zeichen, die mit Nullen gefüllt werden (später wird es mit den empfangenen Daten gefüllt). Zum Lesen der Daten verwenden wir die Methode `radio.read(&text, sizeof(text))`. Dadurch werden die empfangenen Daten in unserem Zeichenarray gespeichert.

Am Ende geben wir einfach die empfangene Nachricht im Serial Monitor aus. Wenn keine Fehler in den Verbindungen vorliegen, sollte die Meldung in Ihrem Serial Monitor wie folgt aussehen:



Es gibt drei weitere Parameter, die wir zwischen `radio.begin` und `radio.openReadingPipe` (oder `openWrittingPipe`) einstellen können. In unseren Beispielsketchen sind diese Parameter auf den Standardwert eingestellt.

Der Erste ist `radio.setChannel(value)`, wobei der "*Wert*" die Kanalnummer im Bereich von 0 bis 125 ist. In unserem Sketch haben wir Kanal 0 in der Funktion `openReadingPipe(0, adress)` verwendet.

Der zweite ist `radio.setPALevel(value)`, wobei der "*Wert*" einer der folgenden Werte ist:

» `RF24_PA_MIN` = -18dBm (default)
» `RF24_PA_LOW` = -12dBm
» `RF24_PA_MED` = -6dBm
» `RF24_PA_HIGH` = 0dBm

Damit stellen wir die von den Modulen verwendete Endstufenstufe ein.

Der dritte ist `radio.setDataRate(value)`, wobei der "*Wert*" einer der folgenden Werte ist:

» `RF24_250KBPS` for 250kbs (default)
» `RF24_1MBPS` for 1Mbps
» `RF24_2MBPS` for 2Mbps

Damit stellen wir die Datenübertragungsrate ein.

Sie haben es geschafft. Sie können jetzt unser Modul für Ihre Projekte nutzen.

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart- Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie noch nach weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei der AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, Ebooks, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us