Willkommen!

Und herzlichen Dank für den Kauf unseres ENC28J60 Netzwerkmoduls für den Arduino. Auf den folgenden Seiten gehen wir mit dir gemeinsam die Anbindung an einen Arduino UNO und ersten Programmierschritte durch.

Viel Spaß!

Das Netzwerkmodul ist ein günstiges, Ethernet basierendes, hoch flexibel programmierbares 10 Mbit Twisted Pair (10 BASE-T) Modul, dessen Herz ein Microchip ENC28J60 ist. Der ENC28J60 Chip besitzt zur Kommunikation mit dem Hostcontroller (unserem Arduino) ein SPI Interface. Über SPI unterstützt der Chip nur den Mode 0, bei SCK (Clock) IDLE Low, LSB First Kommunikation und CS LOW aktiv Status.

Bild 1

Grundsätzlich ist zu beachten, dass der ENC28J60 ein 3,3 Volt Chip ist, dessen Eingänge jedoch 5 Volt tolerant sind und so keinen Levelshifter für den Arduino benötigen. Auch die Ausgänge liegen mit 3,3 Volt Level über der High Detektionsmarke des Arduino und können so grundsätzlich direkt mit unserem Arduino verbunden werden. Dennoch ist es empfehlenswert, für eine stabilere Signalübertragung die Ausgänge mit einem Levelshifter zu verwenden. Wir verzichten in unserer Anleitung zugunsten der Einfachheit jedoch darauf.

Der Controller ist IEEE 802.3 kompatibel und unterstützt halb und voll Duplex bei 10 Mbit mit automatischer Polaritätsdetektion und Korrektur. Er besitzt einen internen 8 KByte großen Sende- und Empfangszwischenspeicher.

Beschaltung

Unser Modul hat insgesamt 12 Anschlüsse. Wenn das Modul wie im Bild 1 gezeigt, vor uns liegt und wir beginnen, von oben links die Reihe hinunter zu zählen und dann nach dem sechsten Pin oben rechts weiter zu zählen, sind diese folgt belegt:

Pin:	Beschreibung:	Zusatzinformationen:
1	5v (V+)	Spannungsversorgung 5 Volt von Arduino
2	LNT (INT)	Interrupt (Kann unbeschaltet bleiben)
3	SO (MISO)	SPI BUS (Master in Slave out)
4	SCK (SCLK)	SPI BUS (BUS Clock - maximal 20 Mhz)
5	RST (Reset)	Chip Reset
6	Q3 (3,3 Volt)	Alternative Spgsvers. max. 3,3 Volt! (Muss unbeschaltet bleiben)
7	GND (Ground)	Spannungsversorgung Masse
8	CLK	Clock out (Kann unbeschaltet bleiben)
9	WOL (Wake on LAN)	Bei entspr. Programmierung können spez. WOL Pakete den Ausgang schalten (Kann unbeschaltet bleiben)
10	ST (MOSI)	SPI BUS (Master out Slave in)
11	CS (Chip Select)	SPI BUS (Chip Select LOW aktiv)
12	GND (Ground)	Spannungsversorgung Masse

Wir verbinden nun unser Modul mit dem Arduino Uno wie folgt:

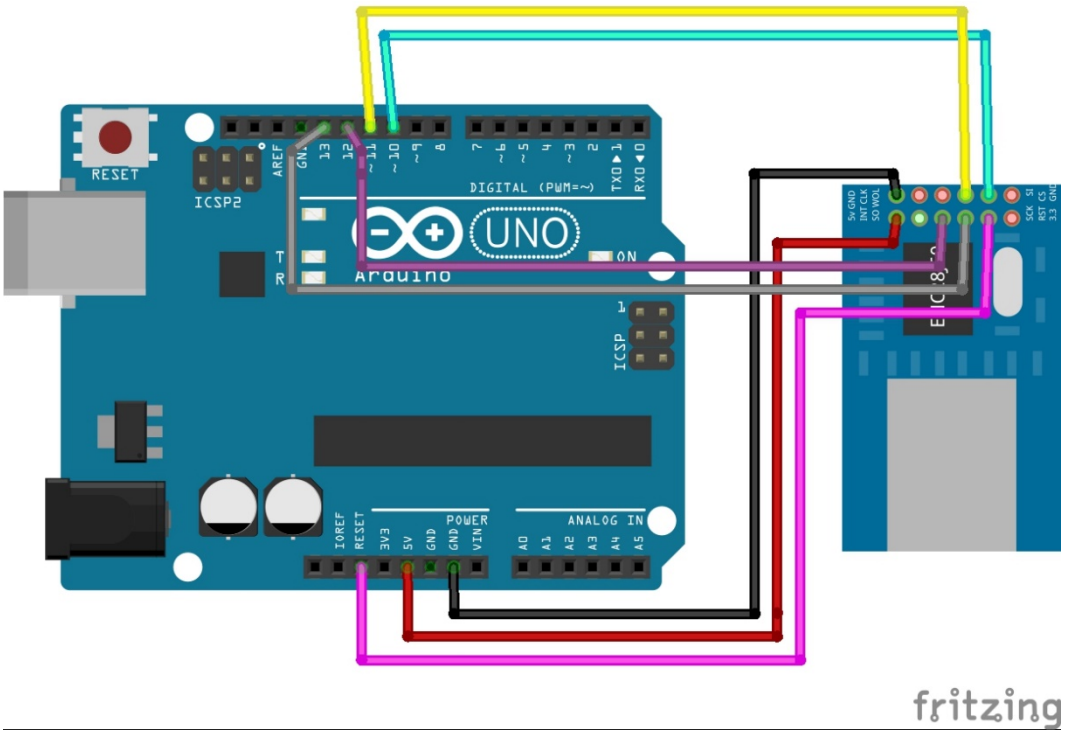

Bild 2 (Verkabelung des Ethernet Modules)

Das Pinmapping seitens des Ardunios ist:

5V	5V	
GND	GND	
SCK	Pin 13	
SO	Pin 12	MISO
ST	Pin 11	MOSI
CS	Pin 10	Änderbar durch die ether.begin() Funktion
RST	RESET	

Damit ist die Grundanbindung an unseren Arduino komplett. Das Modul wird bei jedem Reset unseres Arduinos über die Resetleitung ebenso in einen definierten Ausgangszustand gebracht, sodass hier keine Probleme bei einer evtl. neu Programmierung des Arduinos zu erwarten sind.

Grundsätzliches zur Programmierung

Bevor wir uns eingehender mit der Programmierung des Moduls beschäftigen, müssen wir uns vorab ein paar Dinge über die Arbeitsweise des Ethernet Controllers klar machen:

Der Chip deckt vom OSI Schichten Modell NUR die Schicht1 (Bitübertragung) und Schicht 2 (Sicherung) ab. Informationen zum OSI-Modell finden sich auch auf Wikipedia unter: https://de.wikipedia.org/wiki/OSI-Modell.

Die Parameter (inklusive MAC-Adresse) unter der unser Modul in diesen beiden Schichten arbeitet, können über den SPI Bus als Konfiguration übermittelt werden. Detaillierte Informationen über die Konfigurationsmöglichkeiten des Moduls finden sich im Datenblatt unter:

http://ww1.microchip.com/downloads/en/devicedoc/39662c.pdf

Schicht 3 und alle weiteren folgenden Schichten müssen wir selbst auf unserem Arduino abbilden. Da sich das IP Protokoll in Schicht 3 und das darauf aufsetzende im Netzwerkverkehr am meisten genutzte TCP und UDP Protokoll in Schicht 4 (Transportschicht) befindet, müssen wir auf dem Arduino einen eigenen TCP/IP Stack implementieren. Dies werden wir im nächsten Schritt durch die Einbindung einer entsprechenden Bibliothek auf unserem Arduino tun.

Auch wenn die Implementierung eines eigenen TCP/IP Stacks in unseren Hostcontroller recht aufwendig im ersten Moment scheinen mag und es durchaus auch Netzwerk Module mit integriertem TCP/IP Stack gibt, bietet uns das Modul dafür eine sehr große Flexibilität in Bezug auf Ausgestaltung der Netzwerkkommunikation.

Beispielsweise kann durch die freie Definitionsmöglichkeit des Paket Ether-Typenfeldes mehrere Pakettypen mit dem Modul erstellt oder verarbeitet werden. Siehe dazu: https://de.wikipedia.org/wiki/Ethernet#Das_Typ-Feld_(EtherType)

In unserem Codebeispiel nutzen wir IP- Pakete.

Bibliotheksinstallation

Um nicht komplett einen komplexen TCP/IP Stack für unser Modul selbst programmieren zu müssen, nutzen wir die freie „EtherCard“ Bibliothek für unser Modul. Die „EtherCard“ Bibliothek kann im Arduino Library Manager installiert werden. Alternativ kann es direkt von GitHub heruntergeladen werden.

Dazu erstellen wir in dem Ordner Eigene Dateien/Dokumente/ einen neuen Unterordner mit dem Namen „EtherCard_Library“, und kopieren die Datei EtherCard-master.zip in den neu erstellten Ordner. Die Datei laden wir vorher von GitHub unter der URL https://github.com/njh/EtherCard herunter. (Auf der Webseite auf "Clone or Download" klicken. Dann Download Zip auswählen)

Wir starten wir danach unsere Arduino DIE und binden unter Sketch -> Bibliothek einbinden -> ZIP. Bibliothek einfügen unsere vorher heruntergeladene ZIP Datei ein. Starte die Arduino IDE neu, um die neue "EtherCard"-Bibliothek aktiv werden zu lassen.

Erste Programmierung:

Wir fügen nun in unsere IDE folgenden Arduino Code ein:

```
#include <EtherCard.h>

#define SS 10 // Slave Select Pin Nummer

uint8_t Ethernet::buffer[700]; // Paket Buffer Grüße ist 512 Byte
byte mymac[] =  { 0x74,0x69,0x69,0x2D,0x30,0x31 };// Hier wird die HArdware MAC Adresse definiert.

static BufferFiller bfill;  // used as cursor while filling the buffer

void setup()
{
  Serial.begin(9600); // Öffne serielle Schnittsatelle
  while (!Serial)
    {   // Warte auf seriellen Port
    }
  Serial.println("Warte auf EnC28J60 Startup.");
  delay(6000);
  Serial.println("Initialisierung des Ethernet Controllers");
  if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
    {
     Serial.println( "Fehler: EnC28J60 nicht initalisiert.");
     while (true);
    }
  Serial.println("Hole DHCP Adresse.");
  if (ether.dhcpSetup())
    {
     ether.printIp("IP Adresse: ", ether.myip);
     ether.printIp("Netmask: ", ether.netmask);
     ether.printIp("GW IP: ", ether.gwip);
     ether.printIp("DNS IP: ", ether.dnsip);
    }
  else
    {
     ether.printIp("DHCP Adresse holen ist fehlgeschlagen.");
     while (true);
    }
}

void loop()
{
  word len = ether.packetReceive(); // Paket Listener
  word pos = ether.packetLoop(len);
  if (len)
  {
    Serial.print("IP Packet erhalten. Groesse:");
    Serial.print(len);
    Serial.print(" Bytes. Daten Offset:");
    Serial.print(pos);
    Serial.println(" Bytes. IP Daten:");
    for (int x = 0; x < len; x++)
      {
      char StrC =Ethernet::buffer[x];
      Serial.print(StrC);
      }
    Serial.println("");
  }
}
```

Nachdem wir unser Programm auf den Arduino geladen haben, öffnen wir den seriellen Monitor in unserer IDE und erhalten im Erfolgsfall folgende Ausgabe:

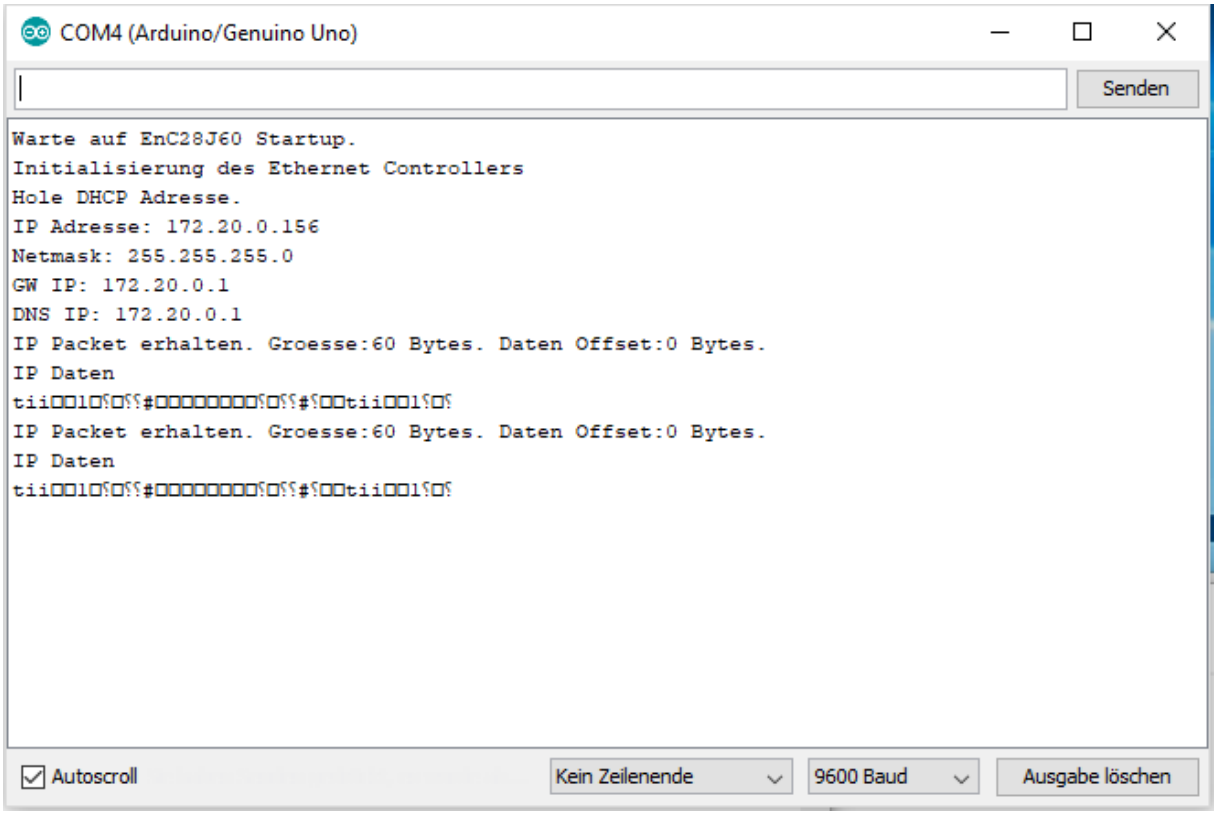

Bild 3

Unser Modul hat die IP Adresse 172.20.0.156 vom DHCP Server neben einem Gateway und einem DNS Server zugewiesen bekommen. (Die Adressen können in deinem Netzwerk abweichen). Wir können nun unser Modul von einem PC aus anpingen:

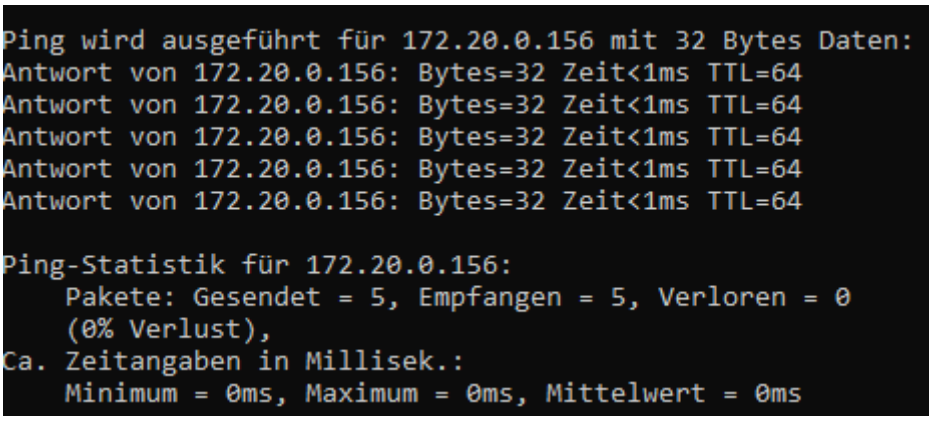

Bild 4 (Ping-Antwort des Moduls)

Wir erhalten eine Antwort von unserem Modul. Auch auf der seriellen Konsole sehen wir den Ping als Datenpaket im Rohformat:

```
IP Packet erhalten. Groesse:74 Bytes. Daten Offset:0 Bytes.
IP Daten
"□2ʕ□tii□□l□E<□+@@□ʕQ□ʕʕ□
OK□□□abcdefghijklmnopqrstuvwabcdefghi
```

Bild 5 (Ping Paket Daten)

Interessant in diesem Zusammenhang ist, dass ein Ping Paket Daten erhalten kann. Windows nutzt als Daten in einem ICMP Ping Paket die wiederholende Zeichenfolge a-w.

Ab jetzt heißt es Experimentieren.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

https://az-delivery.de

Viel Spaß!
Impressum
https://az-delivery.de/pages/about-us