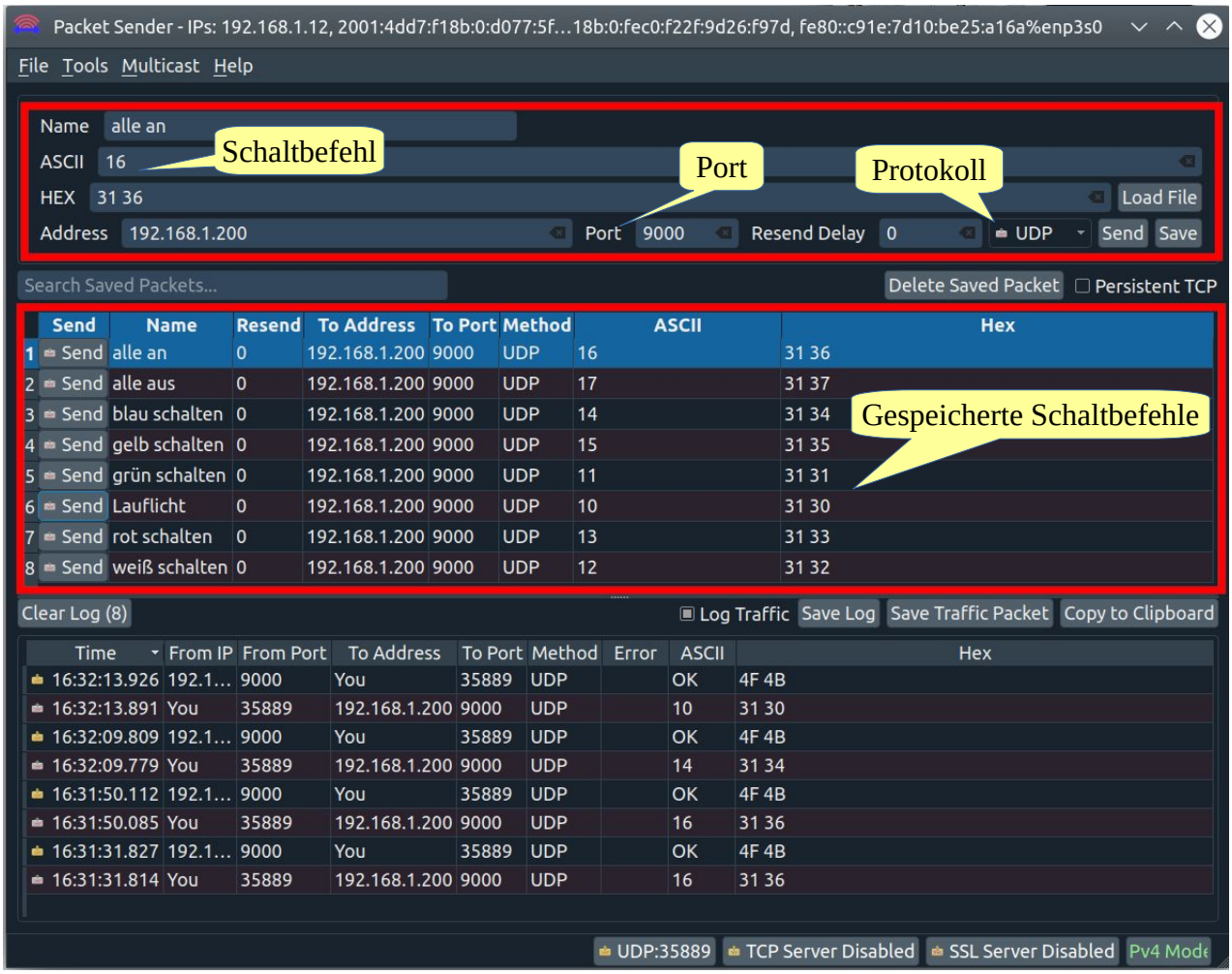
LEDs mit UDP schalten

Verschiedenfarbige LEDs sollen mit Hilfe des UDP-Protokolls im lokalen Netzwerk geschaltet werden. Das User Datagram Protocol (UDP) dient dazu, Nachrichten in einem Netzwerk zu verschicken. Hierzu muss die IP-Adresse des Empfängers und der verwendete Anschluss (Port) bekannt sein.

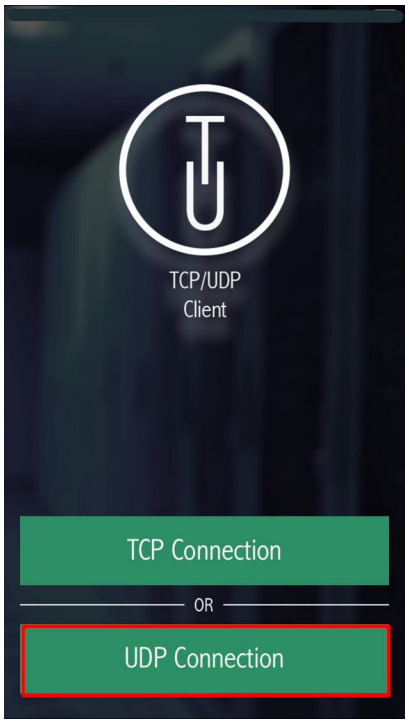
Die Daten werden mit einem entsprechendem Programm oder über eine App übertragen.

Apps	UDP/TCP/Rest Network Utility (iOS) TCP-UDP Client (iOS) UDP-Terminal (iOS Kostenpflichtig) UDP-Sender (Android)
Software	Packet Sender (Windows, Linux, Mac) https://packetsender.com/download

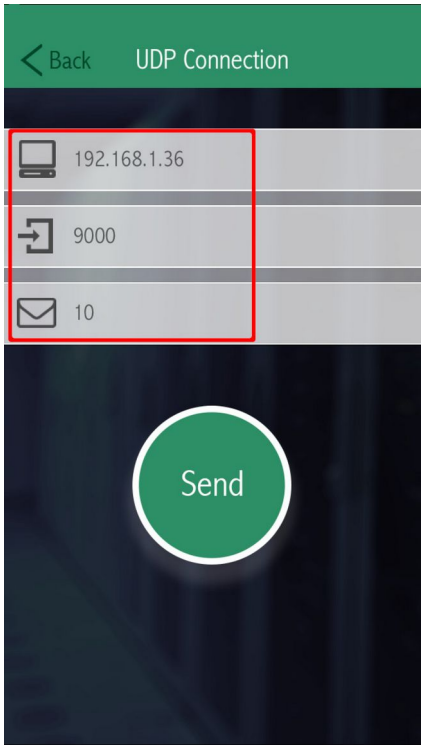
Beispiel: Das Programm Packet Sender



Schalten mit der App TCP-UDP Client



Art der Verbindung wählen



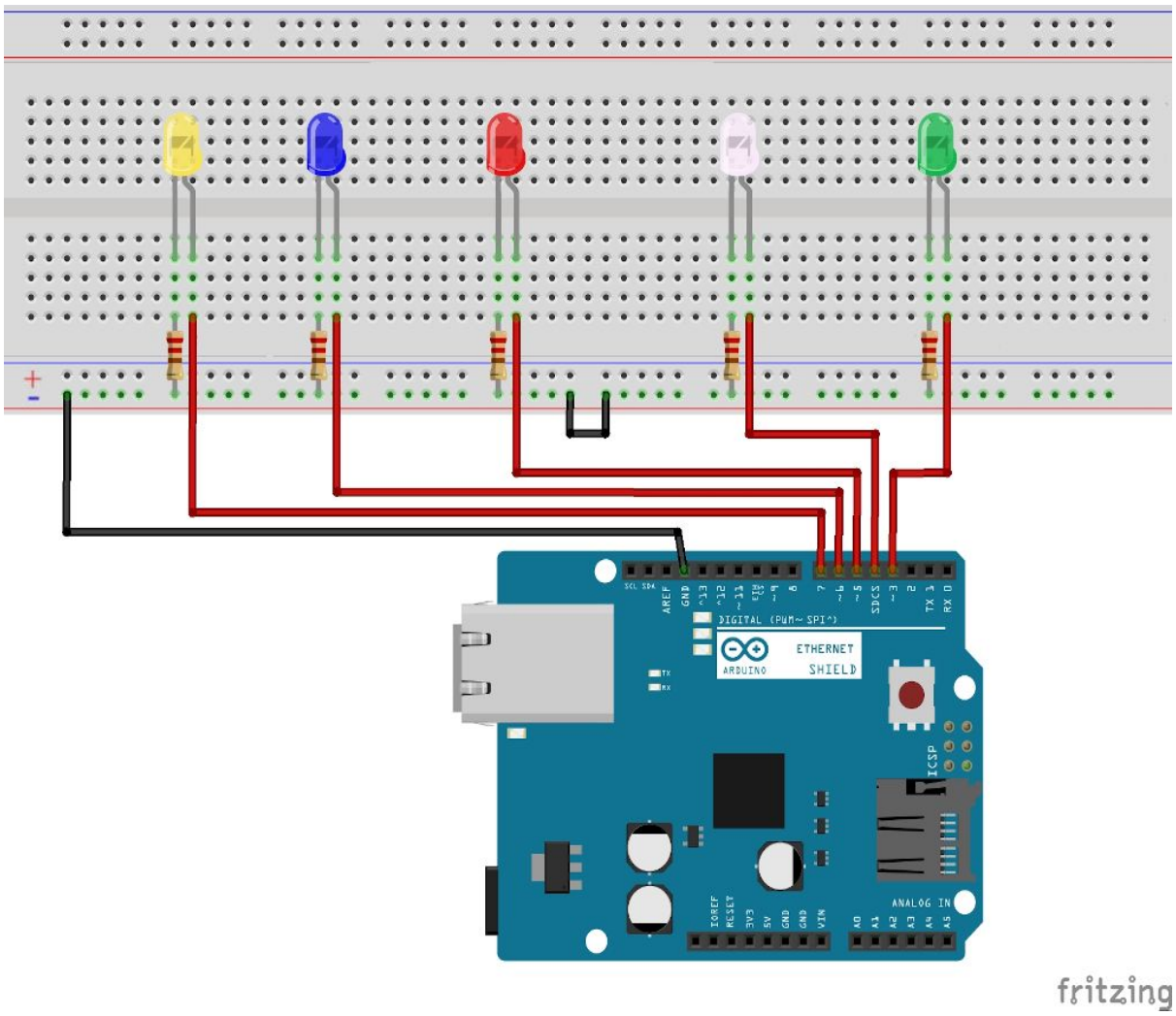
Befehl senden

Für diese Aufgabe benötigst du ein sogenanntes „Shield“, eine Platine, die einfach auf den Arduino aufgesteckt wird. Auf ihr befindet sich ein LAN-Anschluss (RJ45). Alle digitalen und analogen Anschlüsse stehen auch weiterhin zur Verfügung.

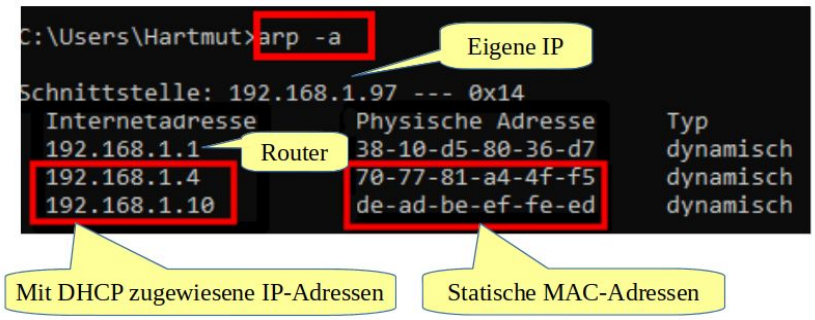
Benötigte Bauteile:

- 5 LEDs
- Ethernet-Shield
- Widerstände> 100 Ω
- Leitungsdrähte

Baue die Schaltung auf.



Für das Programm brauchst du eine freie IP-Adresse und eine freie MAC-Adresse in deinem lokalen Netzwerk.
Du kannst entweder eine feste IP-Adresse vergeben oder die Adresse über DHCP beziehen.
Für die Vergabe eine festen IP kannst du die bereits belegten Adressen mit dem Programm arp herausfinden:



Unter Windows musst du zuerst eine Eingabeaufforderung öffnen.

Mit dem Befehl arp -a kannst du die IP-Adressen und die MAC-Adressen aller in deinem Netzwerk vorhandenen Geräte herausfinden.

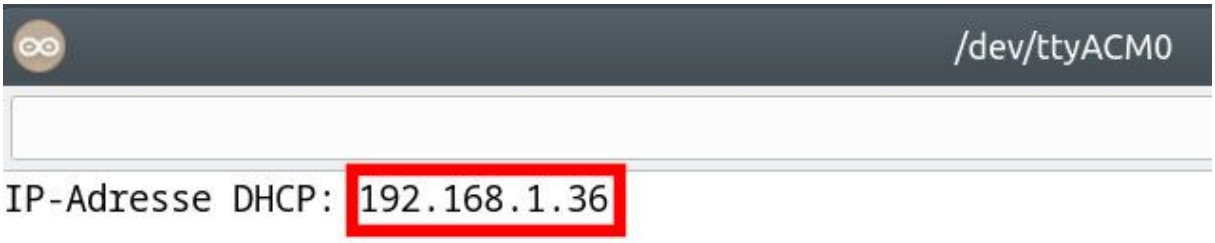
Im Regelfall befindet sich in einem lokalen Netzwerk ein DHCP-Server, der jedem Gerät im Netzwerk automatisch eine IP-Adresse zuteilt. Wenn du dich für diese Alternative entscheidest, musst die im Seriellen Monitor anzeigte Adresse verwenden.

Die MAC-Adresse ist die Hardware-Adresse jeder einzelnen Netzwerkschnittstelle (LAN oder WLAN), mit der jedes Gerät im Netzwerk eindeutig identifiziert werden kann.

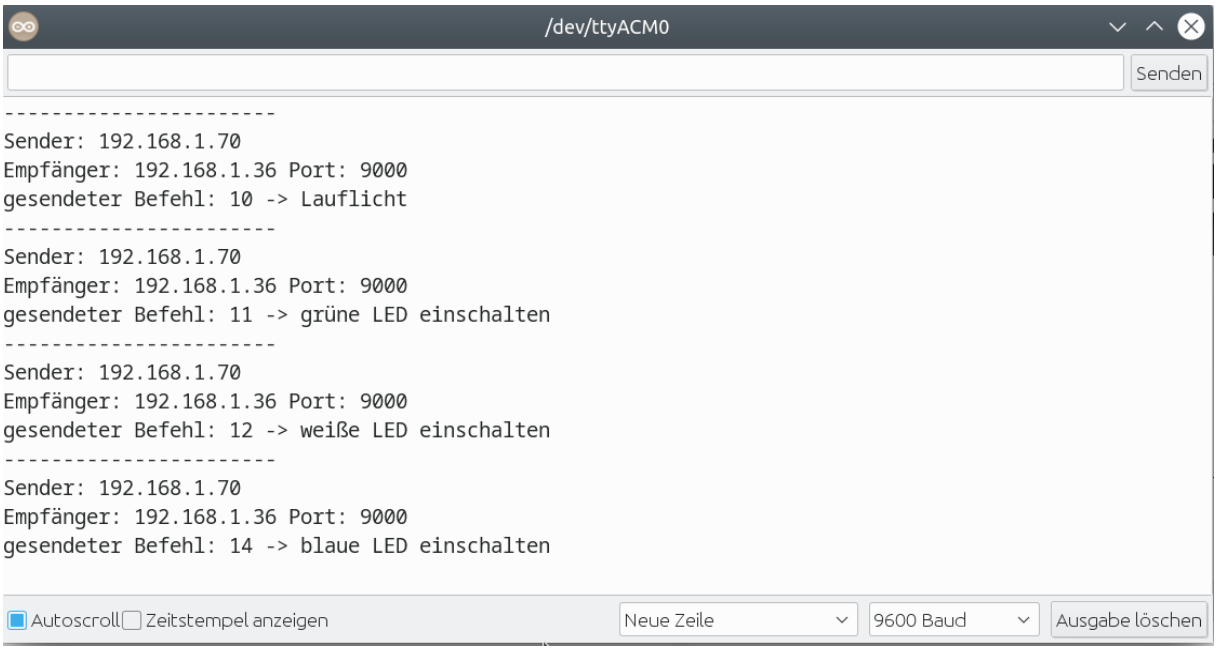
Sie besteht aus sechs Bytes in hexadezimaler Schreibweise, die durch „:“ oder „-“ getrennt werden.

Du kannst die im Programm verwendete „erfundene“ MAC-Adresse übernehmen: Die Gefahr, dass sich ein Gerät mit der gleichen MAC-Adresse im Netzwerk befindet, ist äußerst gering.

Im Seriellen Monitor siehst du die IP-Adresse des Ethernet-Shields:



Im Seriellen Monitor werden die Schaltbefehle angezeigt:



Benötigte Bibliothek:

Sketch → Bibliothek einbinden → Bibliotheken verwalten

Binde die benötigten Bibliotheken ein und definiere die Variablen:

```
# include <Ethernet.h>
# include <EthernetUdp.h>

// die LEDs
enum Farben
{
  GRUEN = 3,
  WEISS,
  ROT,
  BLAU,
  GELB
};

// Status bestimmt, ob die jeweilige LED an oder aus ist
// alle LEDS beim Start aus
bool Status[5] = {0,  0,  0,  0, 0};

// MAC-Adresse und IP definieren
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

/*
  Schalter für die Vergabe der IP
  true -> feste IP
  false IP über DHCP
*/
bool festeIP = false;

// feste IP
IPAddress ip(192, 168, 1, 200);

// lokaler Port
int Port = 9000;

// char-Array für die empfangenen Pakete UDP_TX_PACKET_MAX_SIZE = maximale Größe (24)
char Pakete[UDP_TX_PACKET_MAX_SIZE];

// Variable für den Befehl des Schaltvorgangs
byte Schalten;

// Name für UDP
EthernetUDP Udp;
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  Serial.begin(9600);

  // Ethernet starten feste IP
  if (festeIP) Ethernet.begin(mac, ip);

  // Ethernet starten DHCP
  else Ethernet.begin(mac);
```

```
  // UDP starten
  Udp.begin(Port);

  // pinMode: Startwert → GRUEN, Endwert → GELB
  for (int i = GRUEN; i <= GELB; i++)
  {
    pinMode(i, OUTPUT);
  }
  delay(500);

  if (festeIP) Serial.print("feste IP-Adresse: ");
  else Serial.print("IP-Adresse DHCP: ");
  Serial.println(Ethernet.localIP());
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  // gesendete Pakete abfragen
  UDPAbfragen();

  // Schalten enthält die Anweisung welche LEDs geschaltet werden sollen
  switch (Schalten)
  {
    case 10:
      Serial.print("gesendeter Befehl: ");
      Serial.print(Schalten);
      Serial.println(" -> Lauflicht");
      Lauflicht();
      break;

    case 11:
      // grüne LED
      Status[0] = !Status[0];
      Serial.print("gesendeter Befehl: ");
      Serial.print(Schalten);
      if (!Status[0])
      {
        Serial.println(" -> gr\u00fcne LED ausschalten");
      }
      else Serial.println(" -> gr\u00fcne LED einschalten");

      digitalWrite(GRUEN, Status[0]);
      break;

    case 12:
      // weiße LED
      Status[1] = !Status[1];
      Serial.print("gesendeter Befehl: ");
      Serial.print(Schalten);
```

```
    if (!Status[1])
    {
      Serial.println(" -> wei\u00dfe LED ausschalten");
    }
    else Serial.println(" -> wei\u00dfe LED einschalten");
    digitalWrite(WEISS, Status[1]);
    break;

  case 13:
    // rote LED
    Status[2] = !Status[2];
    Serial.print("gesendeter Befehl: ");
    Serial.print(Schalten);
    if (!Status[2])
    {
      Serial.println(" -> rote LED ausschalten");
    }
    else Serial.println(" -> rote LED einschalten");
    digitalWrite(ROT, Status[2]);
    break;

  case 14:
    // blaue LED
    Status[3] = !Status[3];
    Serial.print("gesendeter Befehl: ");
    Serial.print(Schalten);
    if (!Status[3])
    {
      Serial.println(" -> blaue LED ausschalten");
    }
    else Serial.println(" -> blaue LED einschalten");
    digitalWrite(BLAU, Status[3]);
    break;

  case 15:
    // gelbe LED
    Status[4] = !Status[4];
    Serial.print("gesendeter Befehl: ");
    Serial.print(Schalten);
    if (!Status[4])
    {
      Serial.println(" -> gelbe LED ausschalten");
    }
    else Serial.println(" -> gelbe LED einschalten");
    digitalWrite(GELB, Status[4]);
    break;

  case 16:
    Serial.print("gesendeter Befehl: ");
    Serial.print(Schalten);
    Serial.println(" -> alle einschalten");
    AlleAn();
    break;
```

```
    case 17:
      Serial.print("gesendeter Befehl: ");
      Serial.print(Schalten);
      Serial.println("-> alle ausschalten");
      AlleAus();
      break;

    default:
      break;
  }

  // Schaltbefehl löschen
  Schalten = 0;
}
```

Jetzt fehlen noch die Methoden UDPAbfragen(), Lauflicht(), AlleAn() und AlleAus():

```
void UDPAbfragen()
{
  // Daten abfragen
  int PaketGroesse = Udp.parsePacket();

  // wenn Daten empfangen wurden ...
  if (PaketGroesse)
  {
    // ... Daten lesen
    Udp.read(Pakete, UDP_TX_PACKET_MAX_SIZE);
    Serial.println("----------------------");

    /*
      gesendetes Paket zu int umwandeln
      char-Array in String umwandeln
      zu int umwandeln
      alternativ mit atoi:
      Schalten = atoi(Pakete);
    */
    String SchaltWert = String(Pakete);
    Schalten = SchaltWert.toInt();

    // Schalten = atoi(Pakete);
    // IP des Senders/Port anzeigen
    Serial.print("Sender: ");
    Serial.println(Udp.remoteIP());

    // IP und Port des Ethernet-Shields
    Serial.print("Empf\u00e4nger: ");
    Serial.print(Ethernet.localIP());
    Serial.println(" Port: " + String(Port));

    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write("OK");
    Udp.endPacket();
  }
  delay(10);
}
```

```
void Lauflicht()
{
  AlleAus();
  for (int i = GRUEN; i <= GELB; i++)
  {
    // aktuelle LED i einschalten
    digitalWrite(i, HIGH);
    delay(200);

    // aktuelle LED i ausschalten
    digitalWrite(i, LOW);
  }

  for (int i = GELB; i >= GRUEN; i--)
  {
    // aktuelle LED i einschalten
    digitalWrite(i, HIGH);
    delay(200);

    // aktuelle LED i ausschalten
    digitalWrite(i, LOW);
  }
}
```

```
void AlleAn()
{
  for (int i = GRUEN; i <= GELB; i ++)
  {
    // aktuelle LED i ausschalten
    digitalWrite(i, HIGH);
  }

  // Status für alle LEDs auf 1 setzen
  for (int i = 0; i < sizeof(Status); i++)
  {
    Status[i] = 1;
  }
}
```

```
void AlleAus()
{
  for (int i = GRUEN; i <= GELB; i ++)
  {
    // aktuelle LED i ausschalten
    digitalWrite(i, LOW);
  }

  // Status für alle LEDs auf 0 setzen
  for (int i = 0; i < sizeof(Status); i++)
  {
    Status[i] = 0;
  }
}
```