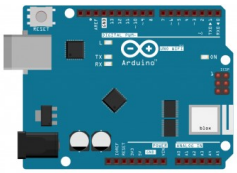
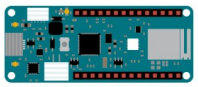
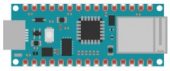
Für diese Anleitung benötigst du einen Arduino mit WiFi.


Arduino WiFi Rev 2


Arduino MKR WiFi 1010

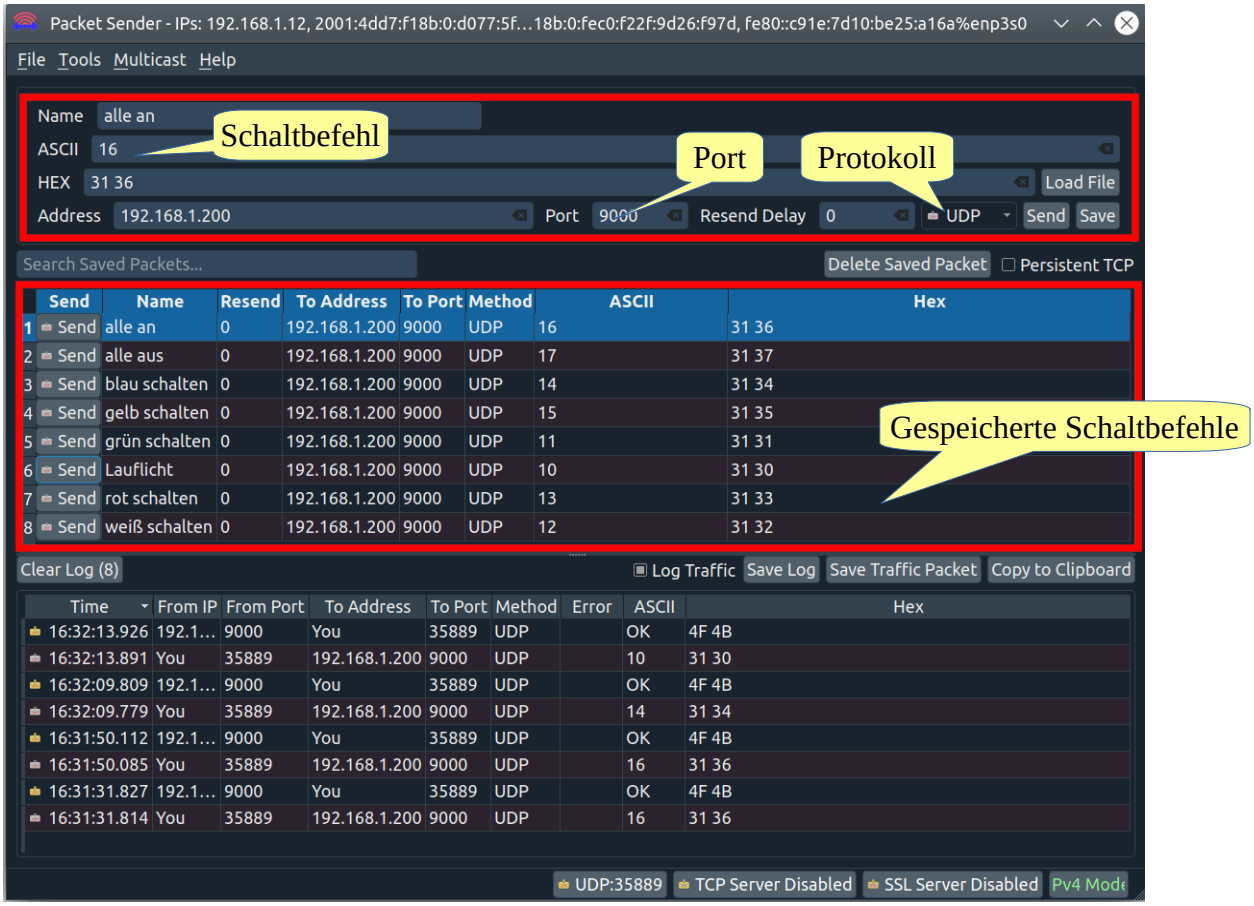

Arduino Nano 33 IoT

Verschiedenfarbige LEDs sollen mit Hilfe des UDP-Protokolls im lokalen Netzwerk über WLAN 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.

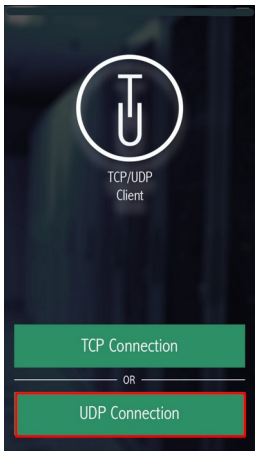
Die Daten werden mit einem entsprechendem Programm oder über eine App übertragen.

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

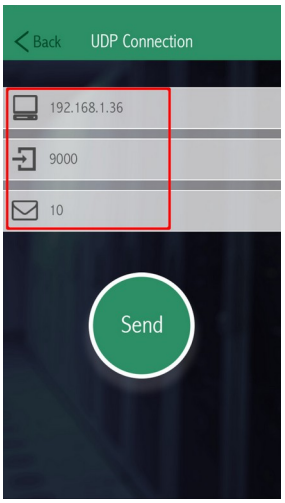
Beispiel: Das Programm Packet Sender



Schalten mit der App TCP-UDP Client



Art der Verbindung wählen

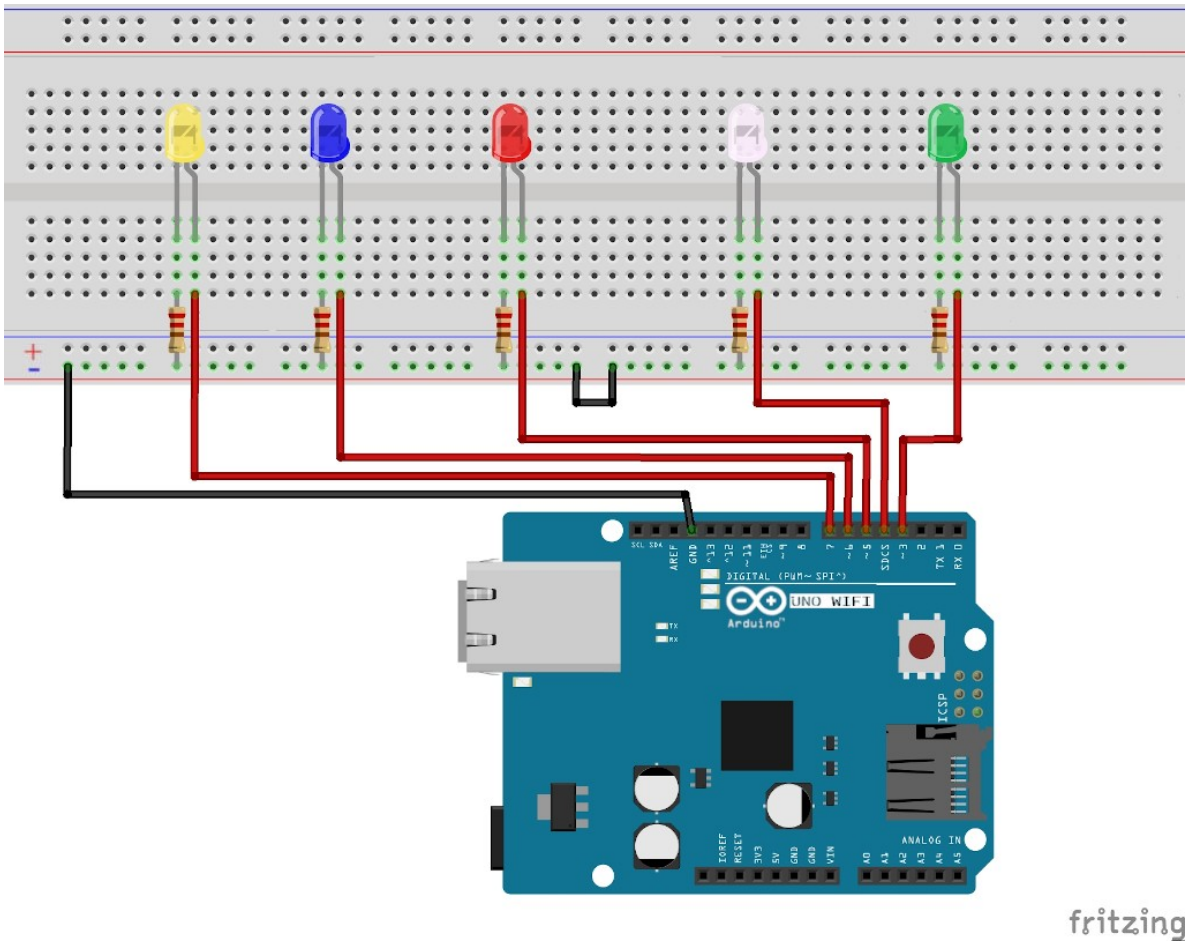


Befehl senden

Benötigte Bauteile:

- 5 LEDs
- 3 Widerstände 220 Ω (gelb, rot und grüne LEDs)
- 2 Widerstände 100 Ω (blaue und weiße LEDs)
- Leitungsdrähte

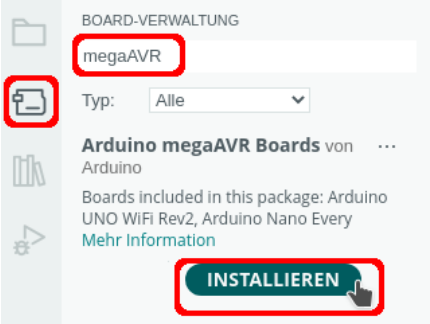
Baue die Schaltung auf.



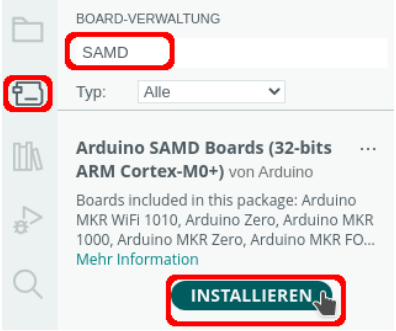
Vorbereitungen:

Board installieren:

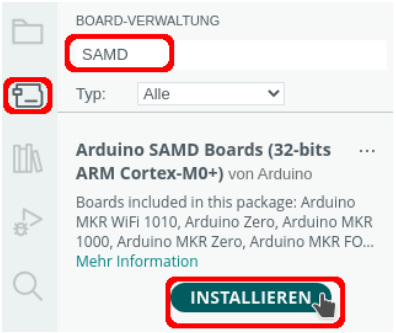
Arduino WiFi Rev 2



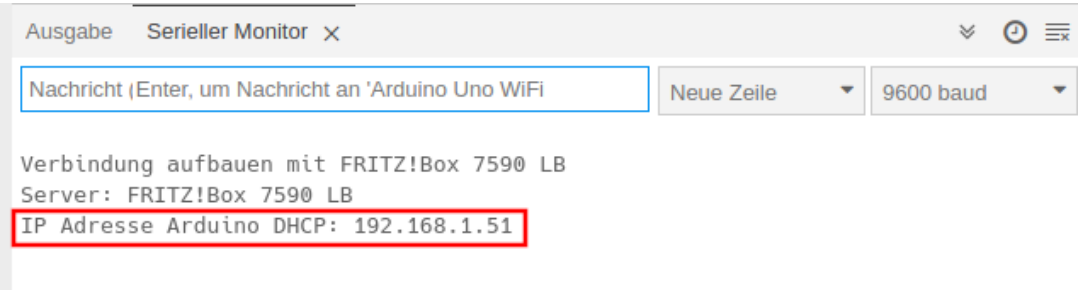
Arduino MKR WiFi 1010



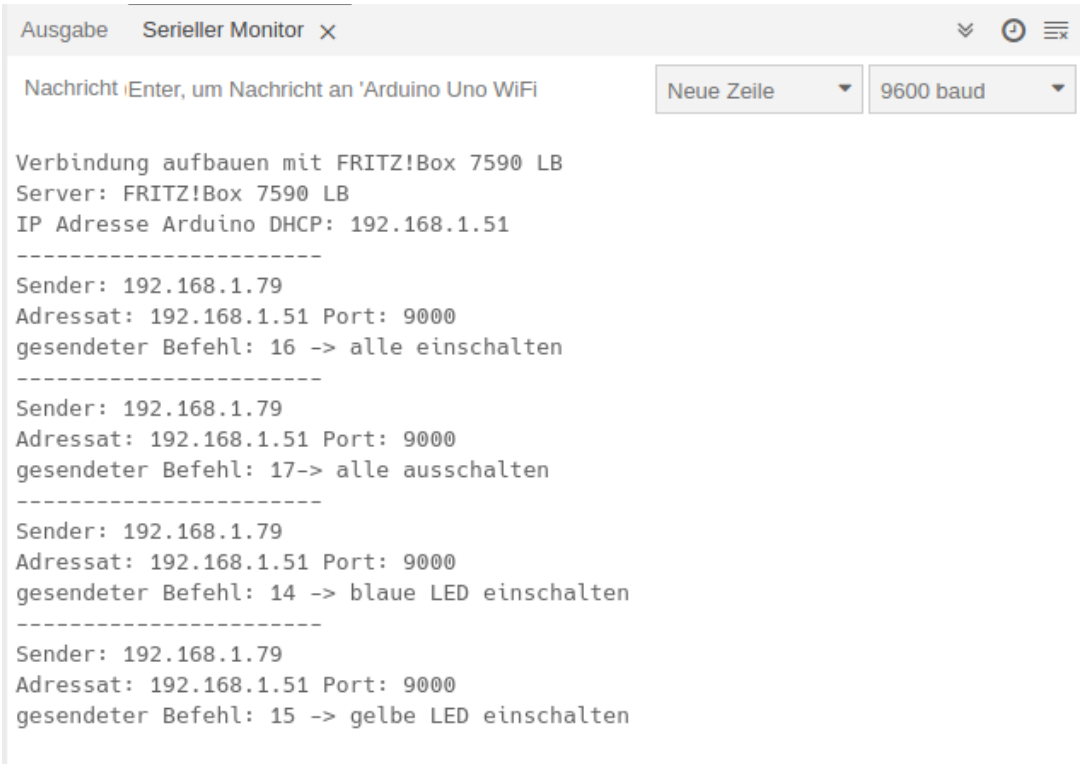
Arduino Nano 33 IoT



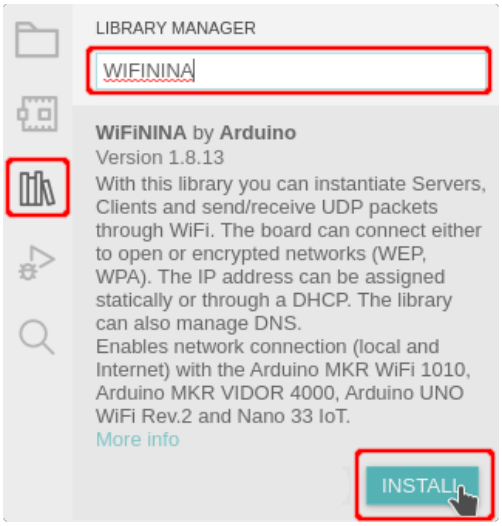
Im Seriellen Monitor siehst du die IP-Adresse des Arduinos:



Im Seriellen Monitor werden die Schaltbefehle angezeigt:



Benötigte Bibliothek:



Binde die benötigten Bibliotheken ein und definiere die Variablen:

```
# include <WiFiNINA.h>
# include <WiFiUdp.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 };

// festeIP = false -> IP-Adresse über DHCP vergeben
// festeIP = true -> IP festlegen
bool festeIP = false;
// feste IP
IPAddress ip(192, 168, 1, 200);
// Netzwerkstatus
int status = WL_IDLE_STATUS;

// Router: Name des Routers
// Passwort: WLAN-Passwort
char Router[] = "FRITZ!Box";
char Passwort[] = "xxxxxxxx";

// lokaler Port
int Port = 9000;

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

```
// Variable für den Befehl des Schaltvorgangs
byte Schalten;

// UDP Namen zuordnen
WiFiUDP Udp;
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  Serial.begin(115200);
  delay(500);

  // statische IP zugeordnet
  if (festeIP) WiFi.config(ip);

  // Verbindung aufbauen
  if (WiFi.status() == WL_NO_MODULE)
  {
    Serial.println(F("Verbindungsaufbau gescheitert!"));
  }

  Serial.print("Verbindung aufbauen mit ");
  Serial.println(Router);

  while (status != WL_CONNECTED)
  {
    status = WiFi.begin(Router, Passwort);

    // Zeit für den Verbindungsaufbau
    // wenn die Verbindung nicht zustandekommt -> Zeit vergrößern
    delay(2000);
  }

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

  // IP des Servers/des verbunden Computers anzeigen
  Serial.print("Server: ");
  Serial.println(WiFi.SSID());

  // IP des Arduinos anzeigen
  if (festeIP) Serial.print("Statische IP Adresse Arduino: ");
  else Serial.print("IP Adresse Arduino DHCP: ");

  // IP des Arduinos anzeigen
  Serial.println(WiFi.localIP());

  Udp.begin(Port);
}
```

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 Funktionen 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("Adressat: ");
    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;
  }
}
```