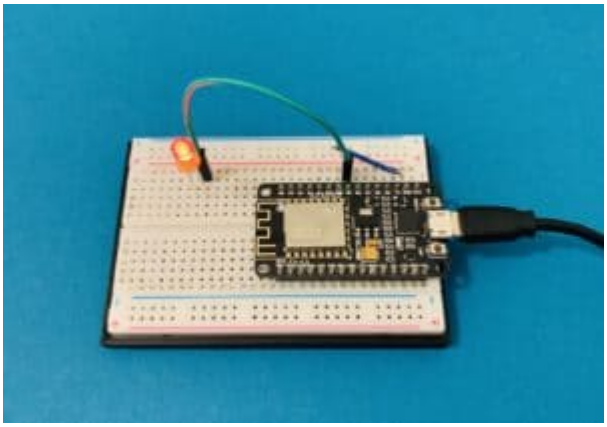
Das Licht ein- und ausschalten mit Telegram



In diesem Projekt lernst du, wie du von deinem Smartphone aus eine Lichtquelle ein- und ausschalten kannst. Außerdem kannst du von dort aus abfragen, ob das Licht gerade an oder aus ist.

Für einen ersten Test verwendest du eine LED, die du an deinem ESP8266 anschließt. Wenn du später möchtest – und vorausgesetzt, du fühlst dich damit sicher – kannst du dich einer „richtigen“ Lampe zuwenden. Hierfür benötigst du dann ein Relais, das du per Telegram steuert und das wiederum den Stromkreis der Lampe öffnet und schließt.

Dieses Projekt ist der vierte Teil einer Serie und baut auf die Vorgängerprojekte auf. In diesem Artikel findest du alles zum Aufbau und den passenden Sketch; wir besprechen jedoch nicht alle Teile des Codes. Wenn du mehr Details erfahren möchtest, wirf bitte einen Blick in die folgenden Projekte:

1. Ein stiller Alarm mit Telegram und einem ESP8266
2. Überwache die Temperatur mit Telegram und einem ESP8266
3. Die aktuelle Temperatur per Telegram abfragen

Falls du noch kein Projekt mit Telegram gebaut hast, lerne zunächst, wie du einen Telegram-Bot erstellst.

Anfänger

1 – 2 Stunden

ca. 10 €

Für dieses Projekt benötigst du (Mengen s. Beschreibung):

[AZDelivery NodeMCU Amica Modul V2 ESP8266 ESP-12F WiFi - Node MCU ESP 8266 WiFi Development Board mit CP2102 kompatibel mit Arduino - inklusive Installationsanleitung als E-Book]
 Maße (LxBxH): 48 x 26 x 13 mm
8,49 €

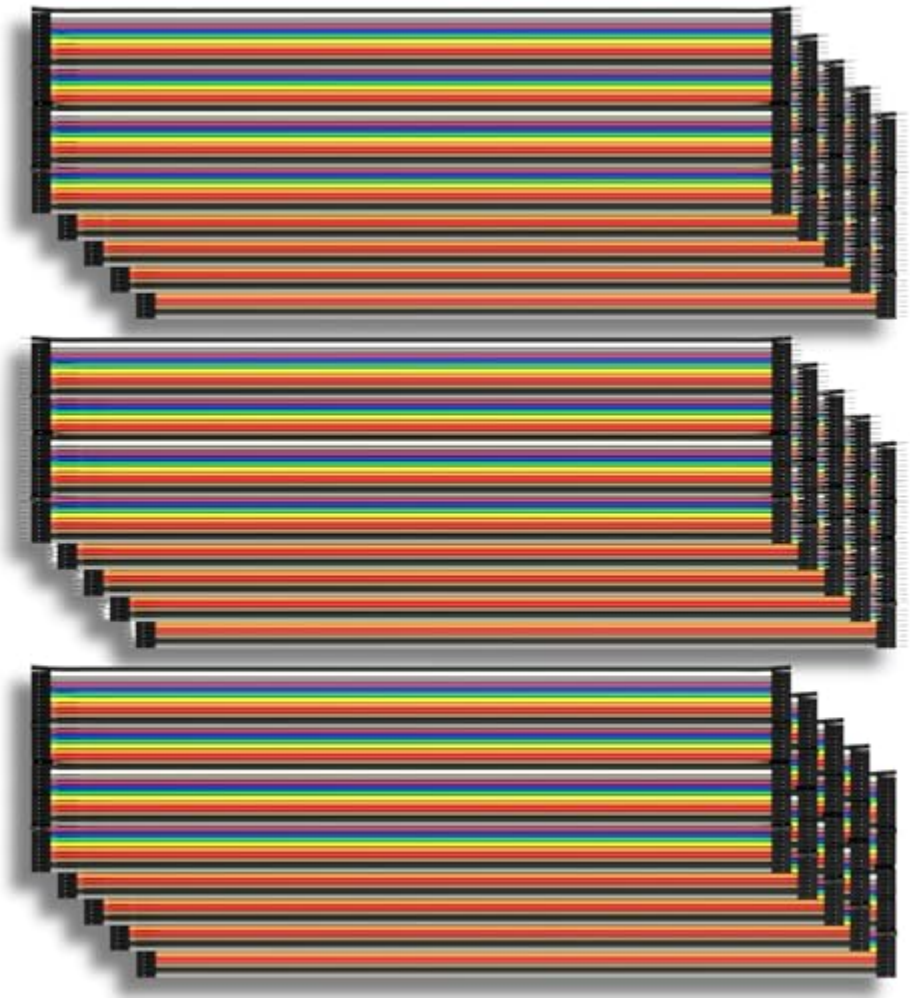
[WayinTop 5mm LED Diode & Widerstände Sortiment Kit für Arduino, 600 Stücke Widerstände von 10 Ohm bis 1 MOhm mit 30 Werten + 200 Stücke 5mm LED Leuchtdiode mit 5 Farben Weiß Rot Blau Grün Gelb]

12,99 €

[AZDelivery Mini Breadboard 400 Pin mit 4 Stromschienen kompatibel mit Arduino und Jumper Wire Kabeln]

 Breadboard; Steckbrett für schnellen Aufbau elektronischer Schaltungen mit 400...

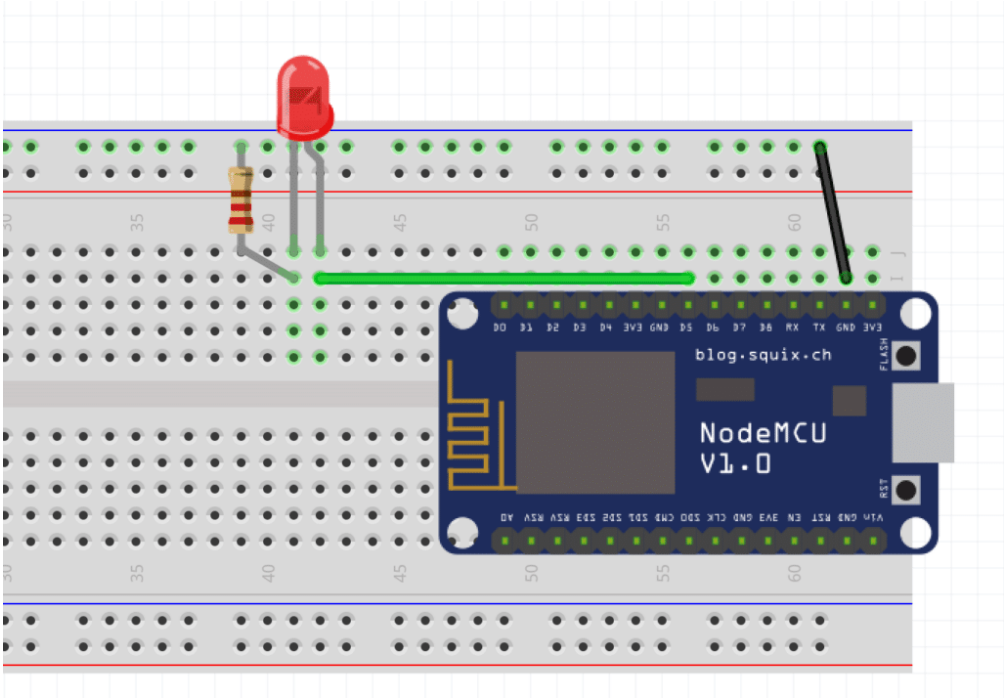
4,99 €



[AZDelivery Jumper Wire Kabel 3 x 40 STK. je 20 cm M2M/ F2M / F2F kompatibel mit Arduino und Raspberry Pi Breadboard inklusive E-Book!]
6,99 €

Der Aufbau des Projekts

Du benötigst für dieses Projekt deinen ESP8266, eine LED samt Widerstand (z.B. 220 Ohm), ein Breadboard und Kabel. Orientiere dich an folgendem Schema:



Wie du oben siehst, verbindest du die Anode (langes Bein) der LED mit dem Pin **D5** an deinem ESP8266. Die Kathode (kurzes Bein) verbindest du über einen Widerstand mit **GND**.

Der Sketch

Kopiere den folgenden Sketch in deine Arduino IDE, ergänze deine Daten und lade ihn auf deinen ESP8266.

Sketch als .txt anschauen

```
/*
    Das Licht ein- und ausschalten mit Telegram - polluxlabs.net
*/

//Bibliotheken
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

// Deine WLAN-Zugangsdaten
const char* ssid = "NETZWERK";
const char* password = "PASSWORT";
```

```
// Den Telegram-Bot initialisieren
#define botToken "TOKEN"  // den Bot-Token bekommst du vom
Botfather)

//Deine UserID
#define userID "USERID"

WiFiClientSecure client;
UniversalTelegramBot bot(botToken, client);

//Variable für das Licht
const int lightPin = 14; //Am ESP8266 Pin D5
bool lightState = LOW;

//Variable für die Anzahl der Anfragen
int numNewRequests;

//Variable für den Text der Anfrage, die du sendest
String text = "";

//UserID des Absenders
String chat_id = "";

//Name des Absenders
String from_name = "";

//Variable für die Willkommensnachricht
String welcome = "";

//Funktion fürs Verarbeiten neuer Anfragen
void handleNewRequests(int numNewRequests) {

  for (int i = 0; i < numNewRequests; i++) { //loopt durch die
neuen Anfragen

     //Checkt, ob du die Anfrage gesendet hast oder jemand
anderes
    chat_id = String(bot.messages[i].chat_id);
    if (chat_id != userID) {
       bot.sendMessage(chat_id, "Du bist nicht autorisiert!",
"");
```

```
      continue;
    }

    // Anfragetext speichern
    text = bot.messages[i].text;
    Serial.println(text);

    from_name = bot.messages[i].from_name;

    if (text == "/start") {
      welcome = "Willkommen, " + from_name + ".\n";
      welcome += "Folgende Befehle kannst du verwenden: \n\n";
      welcome += "/lichtEin \n";
      welcome += "/lichtAus \n";
      welcome += "/status \n";
      bot.sendMessage(chat_id, "http://gph.is/1Rc70ke", "");
      bot.sendMessage(chat_id, welcome, "");
    }

    if (text == "/lichtEin") {
      lightState = HIGH;
      digitalWrite(14, lightState);
      bot.sendMessage(chat_id, "Das Licht ist an.", "");
    }

    if (text == "/lichtAus") {
      lightState = LOW;
      digitalWrite(14, lightState);
      bot.sendMessage(chat_id, "Das Licht ist aus.", "");
    }

    if (text == "/status") {
      if (digitalRead(lightPin)) {
      bot.sendMessage(chat_id, "Das Licht ist an.", "");
      }
      else {
        bot.sendMessage(chat_id, "Das Licht ist aus.", "");
      }
    }
  }
}
```

```
void setup() {

  Serial.begin(115200);
  client.setInsecure();

  pinMode(lightPin, OUTPUT);
  digitalWrite(lightPin, lightState);

  //Verbindung zum WLAN
  Serial.print("Verbinde mich mit: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println("");
  Serial.println("Verbunden!");
}

void loop() {

  //checkt, ob eine neue Anfrage reinkam
                    int          numNewRequests          =
bot.getUpdates(bot.last_message_received + 1);

   while (numNewRequests) { //wird ausgeführt, wenn
numNewRequests == 1
    Serial.println("Anfrage erhalten");
    handleNewRequests(numNewRequests);
     numNewRequests = bot.getUpdates(bot.last_message_received
+ 1);
  }
  delay(1000);
}
```

Was ist neu in diesem Sketch?

Viel musst du in deinem Code eigentlich nicht verändern, um statt der Temperaturabfrage das Licht ein- und auszuschalten. Werfen wir einen Blick auf die Details.

Zunächst benötigst du eine Konstante für den Pin, an dem die LED angeschlossen ist und eine Variable für den Zustand der LED – also ob sie an oder aus ist.

```
const int lightPin = 14; //Am ESP8266 Pin D5
bool lightState = LOW;
```

In der Setup-Funktion legst du den **pinMode** für den LED-Pin fest und weist der LED den Zustand zu, der in der Variablen **lightState** steht – also zu Beginn **LOW** – aus.

```
pinMode(lightPin, OUTPUT);
digitalWrite(lightPin, lightState);
```

Die Funktion handleNewRequests()

Diese Funktion kennst du ja bereits aus dem vorangegangenen Projekt. Hier werden deine Anfragen verarbeitet, die du von deinem Smartphone aus sendest.

Du kannst auch GIFs nutzen, sende dazu einfach die entsprechende URL. Im Sketch oben befindet sich schon eine.

In diesem Projekt gibt es drei Anfragen (oder Befehle), die du senden kannst: **/lichtEin**, **/lichtAus** und **/status**. Mit der letzten Anfrage fragst du ab, ob die LED gerade ein- oder ausgeschaltet ist und erhältst die entsprechende Antwort zurück.

Wenn du also deinem ESP8266 den Text **/lichtEin** sendest, setzt dieser die Variable **lightState** auf **HIGH** und schaltet die LED mit **digitalWrite()** ein. Zuletzt sendet er dir mit der bekannten Funktion **bot.sendMessage()** Feedback.

```
if (text == "/lichtEin") {
  lightState = HIGH;
```

```
  digitalWrite(14, lightState);
  bot.sendMessage(chat_id, "Das Licht ist an.", "");
}
```

Der Befehl **/lichtAus** funktioniert so ähnlich, nur dass er die LED natürlich ausschaltet. Wenn du **/status** sendest, prüft dein ESP8266, ob die LED ein- oder ausgeschaltet ist:

```
if (text == "/status") {
  if (digitalRead(lightPin)) {
  bot.sendMessage(chat_id, "Das Licht ist an.", "");
  }
  else {
    bot.sendMessage(chat_id, "Das Licht ist aus.", "");
  }
}
```

Wie geht es weiter?

Wie eingangs erwähnt, kannst du statt der LED ein Relais steuern. Je nachdem, welches Signal du diesem von deinem Smartphone aus sendest, öffnet und schließt das dann den Stromkreis eines anderen Geräts – z.B. einer Lampe oder einer Kaffeemaschine.