

Grundlagen Programmierung

Tobias Weckerle, HR/AUSB

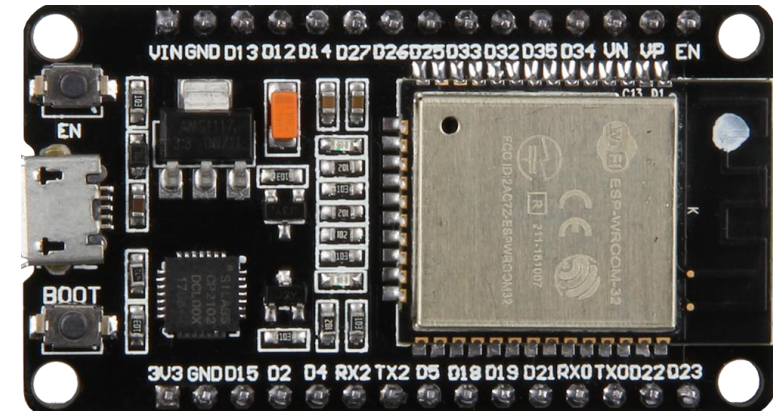


Agenda

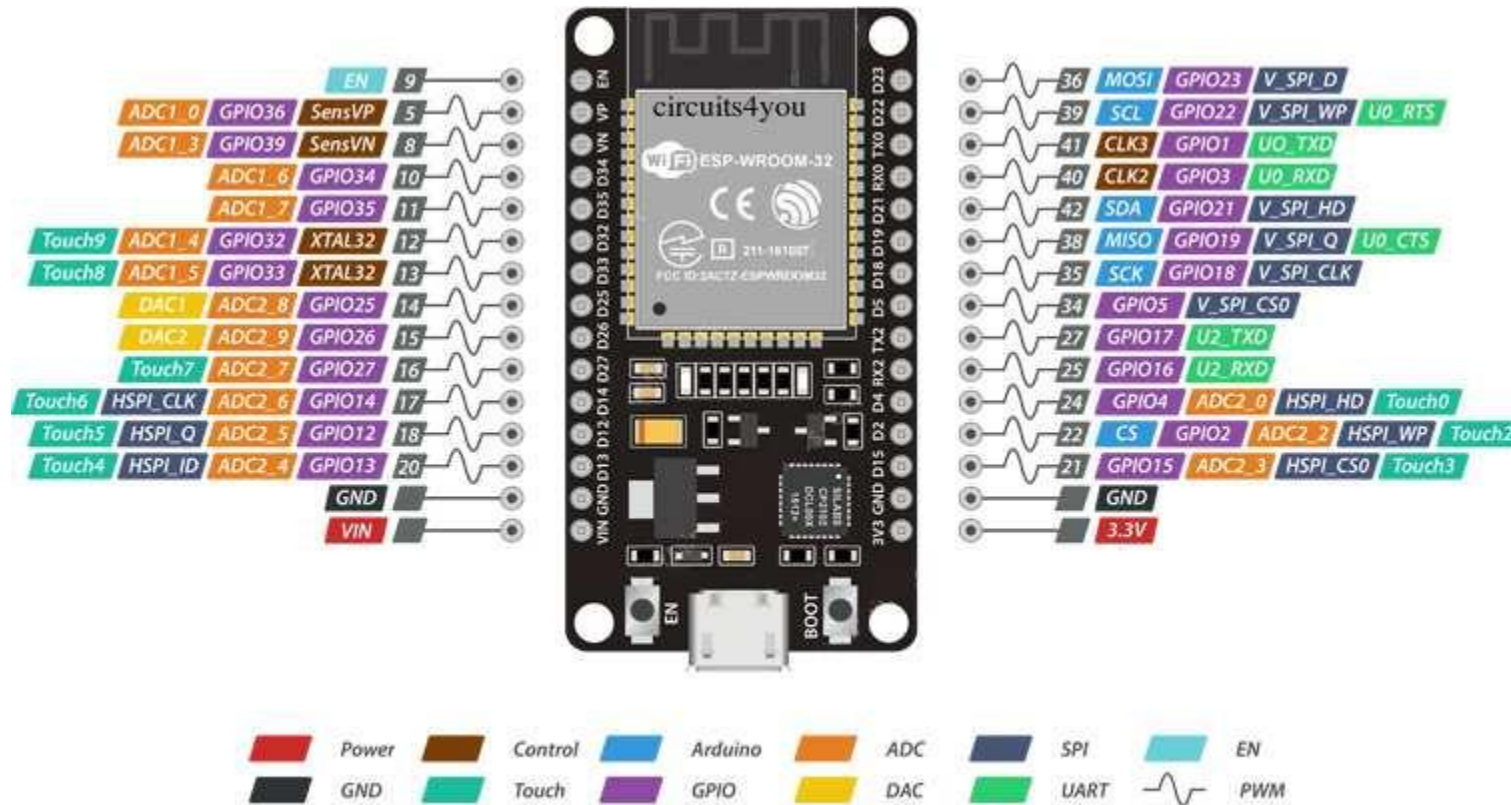
1. NodeMCU
 2. Pinout
 3. Die Arduino IDE
 4. Hallo Welt
 5. Grundlagen
 6. Inputs und Outputs
 7. Ansteuerung von Hardware
 8. Tipps und Tricks
1. Jetzt wird es smart
 - a. Aufbau
 - b. Netzwerk
 - c. MQTT
 - d. NodeRed
 - e. Daten senden über NodeMCU

Unsere intelligente Steuerung

- › Programmierung über Arduino IDE
- › 2,4 GHz Dual-Mode Wifi
- › BT-Funkverbindung
- › 512 kB Arbeitsspeicher
- › 4MB Speicher
- › 25 GPIOs
- › Peripherie (ADC / UART / SPI / I2C / PWM)



Pinout



ESP32 Dev. Board Pinout

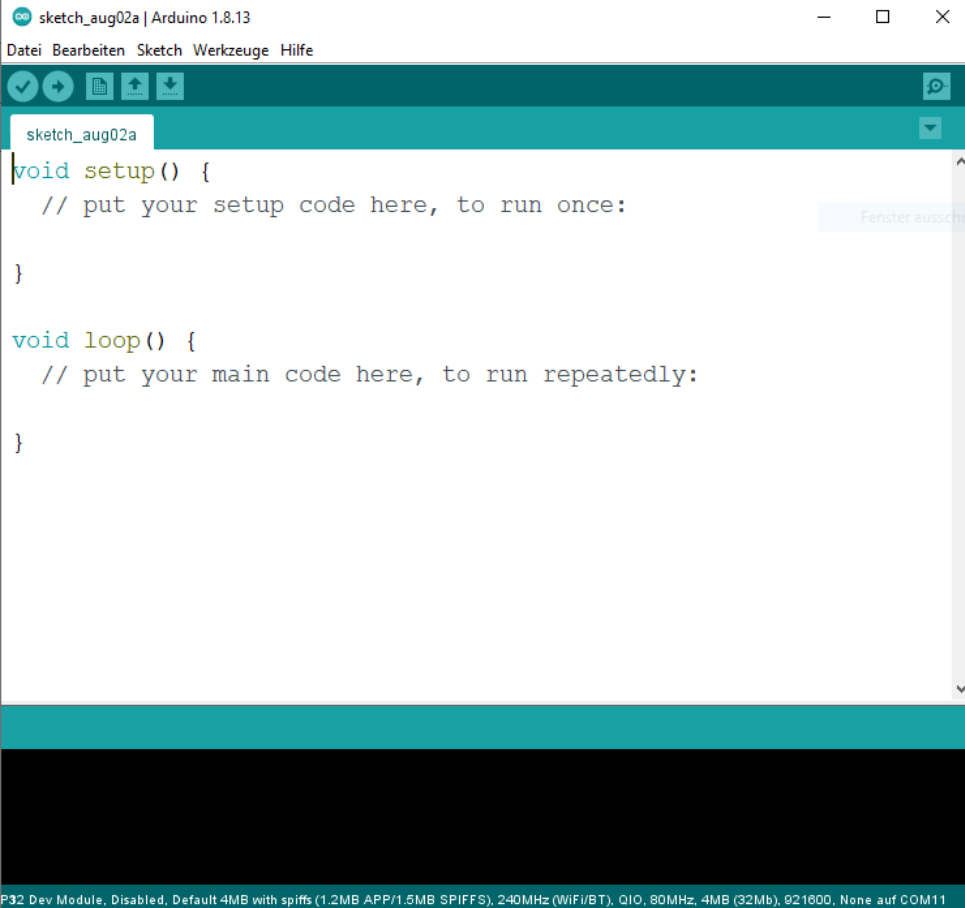
Die Arduino IDE

Die setup() Methode

- › Wird zu Beginn aufgerufen
- › Wird genau einmal ausgeführt

Die loop() Methode

- › Wird nach der Setup aufgerufen
- › Wird unendlich oft ausgeführt



```
sketch_aug02a | Arduino 1.8.13
Datei Bearbeiten Sketch Werkzeuge Hilfe

sketch_aug02a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

P32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None auf COM11
```

- › Onboard LED auf dem NodeMCU blinken lassen und seriell „Hallo Welt“ schreiben

```
#define LED_BUILTIN 2

void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  Serial.println("Hallo Welt");
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Mit der delay() Methode wartet der NodeMCU Mikrocontroller

- › Die Zeit wird in Millisekunden angegebende
- › 1000 für eine Sekunde
- › Programm bleibt an der Stelle stehen und macht nichts

Serielle Kommunikation

Kommunikation über UART

- › Mit dem Seriellen Monitor kannst du Daten senden und empfangen

```
void setup() {  
    // put your setup code here, to run once:  
    Serial.begin(9600);  
    Serial.println("Setup");  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    Serial.println("Loop");  
  
    //Wait for 1 second  
    delay(1000);  
}
```


- › Mit Variablen könne wir Zeichen, Zahlen, Texte und Wahrheitswerte speichern

```
char myCharacter = 'a';
```

```
int myNumber = 5;
```

```
float myFloat = 23.4f;
```

```
double myDouble = 22.2;
```

```
string myText = text;
```

```
bool myBoolean = true;
```

Operatoren und deren Verwendung

- › Wertzuweisung (=)
- › ODER (|)
- › UND (&)
- › Rechenoperationen (+, -, *, /, %)
- › Vergleichsoperatoren
 - Gleichheit (==)
 - Ungleichheit (!=)
 - Größer, kleiner, größer gleich, kleiner gleich (>, <, >=, <=)

Eine Schleife hilft dabei wiederholende Tätigkeiten auszuführen

› Zählerbasierte Schleife

```
for(int i=0; i<10; i++)  
{  
    Serial.print("Zahl");  
    Serial.println(i);  
}
```

› Bedingungsbasierte Schleife

```
while(i<5)  
{  
    Serial.print("Zahl");  
    Serial.println(i);  
    i++;  
}
```

Bedingungen

Bedingungen mit if

Mit Bedingungen kann zwischen ein oder mehreren Möglichkeiten entschieden werden

- › If, else if und else

```
int age = 17;

if (age < 17) {
    Serial.println("Du bist jünger als 17 Jahre");
}

else if (age > 17) {
    Serial.println("Du bist älter als 17 Jahre");
}

else {
    Serial.println("Du bist bist 17 Jahre alt");
}
```

Bedingungen

Bedingungen mit switch

Das switch-Statement kann mehrere Wege unterscheiden

› Switch

```
int age = 17;

switch (age)
{
    case 17:
        Serial.println("Du bist bist 17 Jahre alt");
        break;
    default:
        Serial.println("Du bist nicht 17 Jahre alt");
        break;
}
```

Funktionen können wiederkehrende Inhalte haben und beliebig oft und verschachtelt aufgerufen werden

- › Funktionen können einen Rückgabewert haben
- › Funktionen können Parameter haben
- › Funktionen bestehen aus zwei Teilen
 - Definition der Funktion
 - Aufruf der Funktion

Funktion ohne Parameter

› Definition

```
void sayHello()  
{  
  Serial.print("Hallo");  
}
```

› Aufruf

```
sayHello();
```

Funktion mit Parameter

› Definition

```
void sayName(String myName)
{
    Serial.print(myName);
}
```

› Aufruf

```
sayName(„Erwin“);
```


Funktion mit Rückgabewert

› Definition

```
String getName()  
{  
    return "Erwin";  
}
```

› Aufruf

```
String erwin = getName();
```

Mehrere Werte vom gleichen Typ können in einem Array gespeichert werden

- › Beispiel mit einem Array von Zahlen, welches durch eine foreach Schleife ausgegeben wird

```
void loop()
{

    int numbers[] = {1, 7, 4, 2};

    //for each integer number in array numbers
    for (int number : numbers)
    {
        Serial.println(number);
    }
    delay(1000);
}
```

Es ist möglich digitale Ein- und Ausgänge des Mikrocontrollers zu steuern

- › Digitale Pins können über HIGH und LOW angesteuert und ausgelesen werden

```
void setup() {  
    pinMode(18, INPUT);  
    pinMode(19, OUTPUT);  
}  
  
void loop()  
{  
    int val = digitalRead(18);  
    if(val==HIGH){ digitalWrite(19, HIGH);}  
    else{digitalWrite(19,LOW);}  
}
```

Es ist möglich analoge Ein- und Ausgänge des Mikrocontrollers zu steuern

- › Analoge Pins können einen Spannungswert (z.B. 0-255) lesen oder schreiben
- › ACHTUNG: NodeMCU unterstützt die Methode analogWrite() nicht

```
void setup() {  
    pinMode(18, INPUT);  
    pinMode(19, OUTPUT);  
    sigmaDeltaSetup(0, 312500);  
    sigmaDeltaAttachPin(19,0);  
    sigmaDeltaWrite(0, 0);  
}
```

```
void loop() {  
    int val = analogRead(18);  
    sigmaDeltaWrite(0,val);  
}
```

Einfache Eingänge und Ausgänge

- › Motor-Shield (I/O)
- › Bewegungsmelder (I/O)

Arbeiten mit Bibliotheken

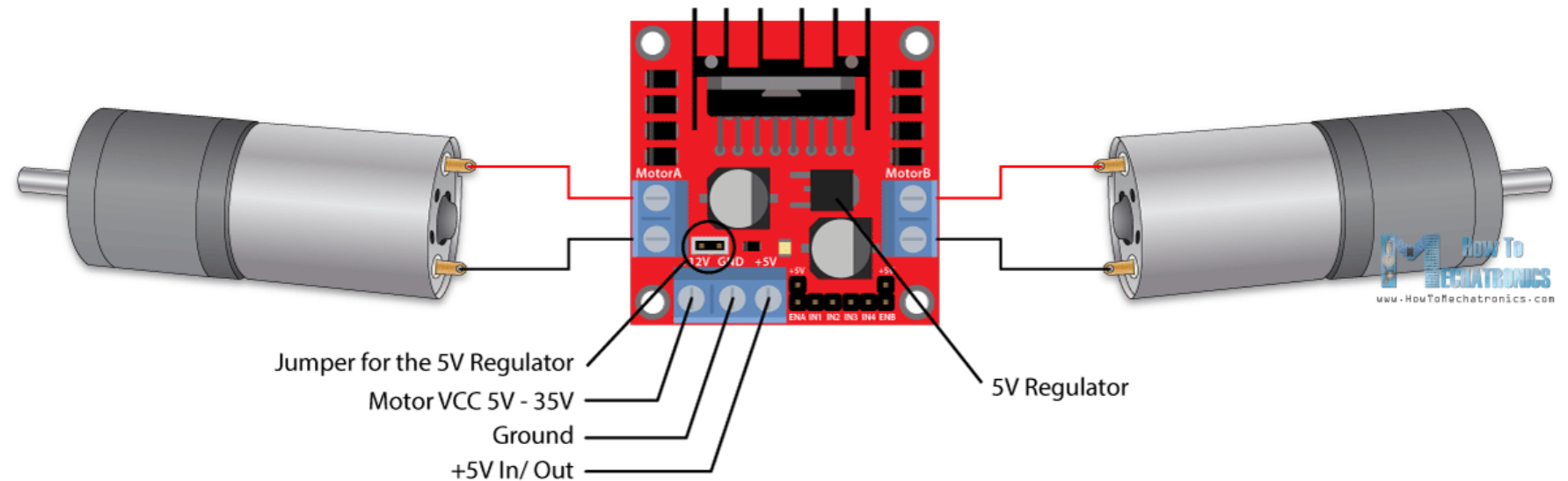
- › RGB-LED (Neopixel Library)
- › Temperatur auslesen

Bewegungsmelder

```
void setup() {  
    pinMode(18, INPUT);  
    Serial.begin(9600);  
}  
  
void loop(){  
    if (digitalRead(18) == HIGH) {  
        Serial.println("Bewegung erkannt");  
    }  
    else {  
        Serial.println("keine Bewegung erkannt");  
    }  
    delay(500);  
}
```

Der Motor1 kann über die Pins In1, In2 und ENA angesteuert werden

- › Vorwärts (in1 = HIGH, in2 = LOW)
- › Rückwärts (in1 = LOW, in2 = HIGH)
- › Stop (in1 = HIGH, in2 = HIGH)
- › Speed (Analogwert zwischen 0 und 255 auf ENA)



Über eine Bibliothek (Neopixel Library) können die LEDs angesteuert werden

› Global

```
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRBW + NEO_KHZ800);
```

› Setup

```
pixels.begin();
```

› Loop

```
pixels.clear(); // Set all pixel colors to 'off'  
pixels.setPixelColor(i, pixels.Color(0, 255, 0)); //Set pixel i to green  
pixels.show(); // Send the updated pixel colors to the hardware.
```


- › Immer an das **Semikolon** denken
- › Funktionen zur **Strukturierung** nutzen
- › Passende Wahl der **Variablennamen**
- › **Kommentare** ergänzen
- › Immer wieder eine **Version** abspeichern
- › Serial.print zum **Debuggen** verwenden
- › **Gefahren**, wenn == oder = verwendet wird
- › **Geschützte Wörter** vermeiden (void, if, name, ...)
- › In **kleinen Schritten** entwickeln
- › Viel **Testen**
- › Konkrete **Planung**, was programmiert werden soll
- › **Hilfe** holen, wenn ihr nicht weiterkommt ;-)

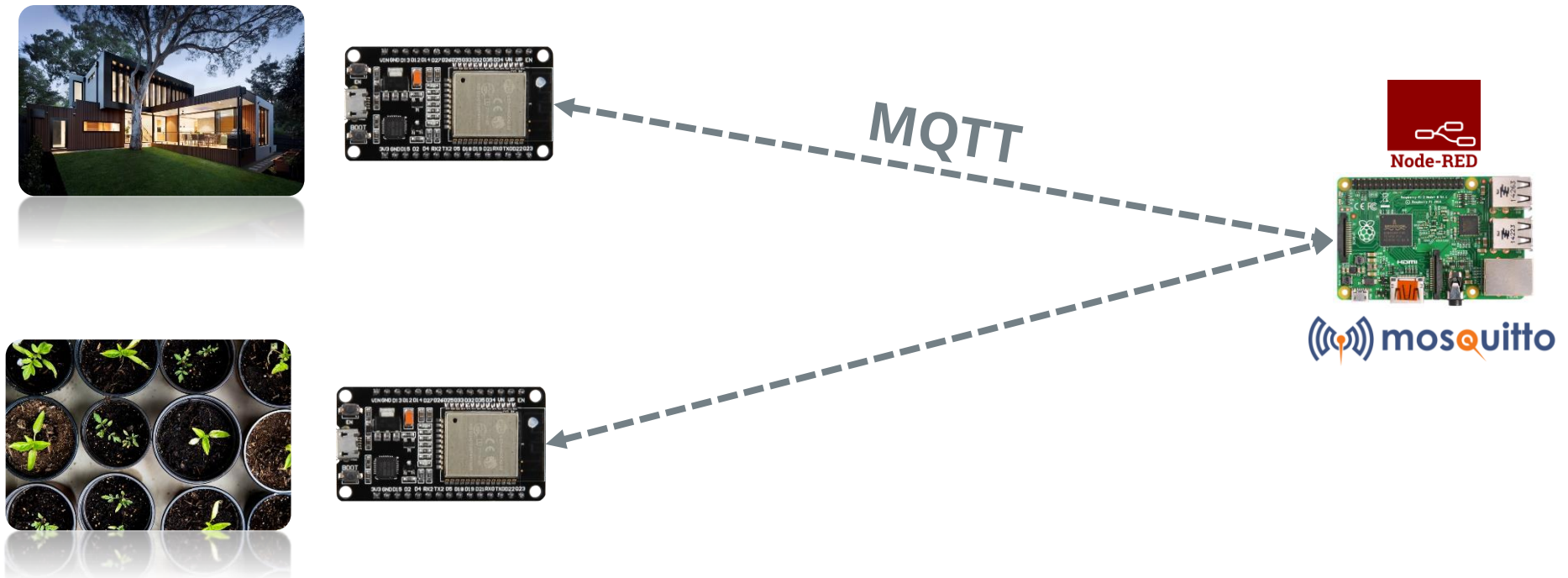


Jetzt wird es smart!

Weiter geht es mit MQTT und NodeRed

Aufbau

Wie können wir Daten zwischen NodeMCU und RaspberryPi austauschen?



Zentraler Router

- › IP-Adresse: 192.168.8.1
- › WLAN: SICK Summer University
- › PASSWORT: SSU 2021!

Gruppe 1:

- › Raspberry: 192.168.8.11
- › Haus: automatisch (DHCP)
- › Garten: automatisch (DHCP)

(NodeRed: <http://192.168.8.11:1880>, UI: <http://192.168.8.11:1880/ui>)

Gruppe 2:

- › Raspberry: 192.168.8.21
- › Haus: automatisch (DHCP)
- › Garten: automatisch (DHCP)

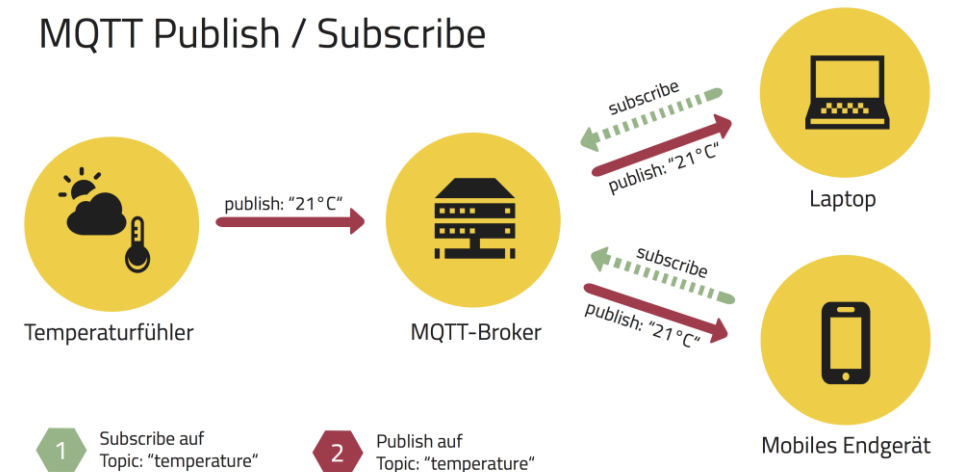
(NodeRed: <http://192.168.8.21:1880>, UI: <http://192.168.8.21:1880/ui>)

Wie funktioniert MQTT?

Message Queuing Telemetry Transport (MQTT)

- › Netzwerkprotokoll (Machine-to-Machine – M2M)
 - › Ein zentraler Vermittler (Broker) koordiniert die Nachrichten
 - › Nachrichten können gesendet werden (Publish)
 - › Nachrichten können gelesen werden (Subscribe)
 - › Die Unterhaltung findet auf einem Kanal (Topic) statt
-
- › Beispiel
 - Arduino hört auf Kanal „home/helligkeit“
 - Dashboard sendet gewünschte Helligkeit an „home/helligkeit“

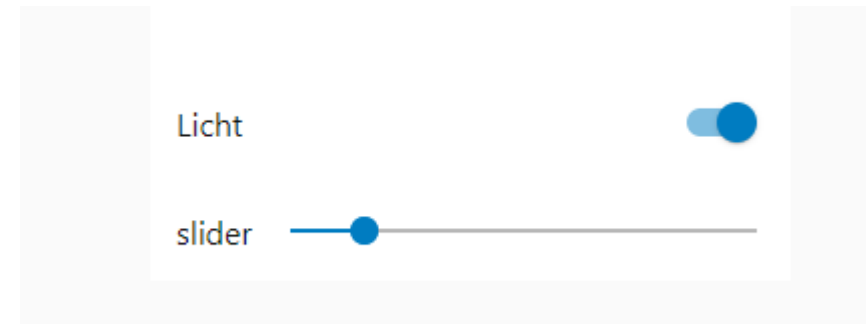
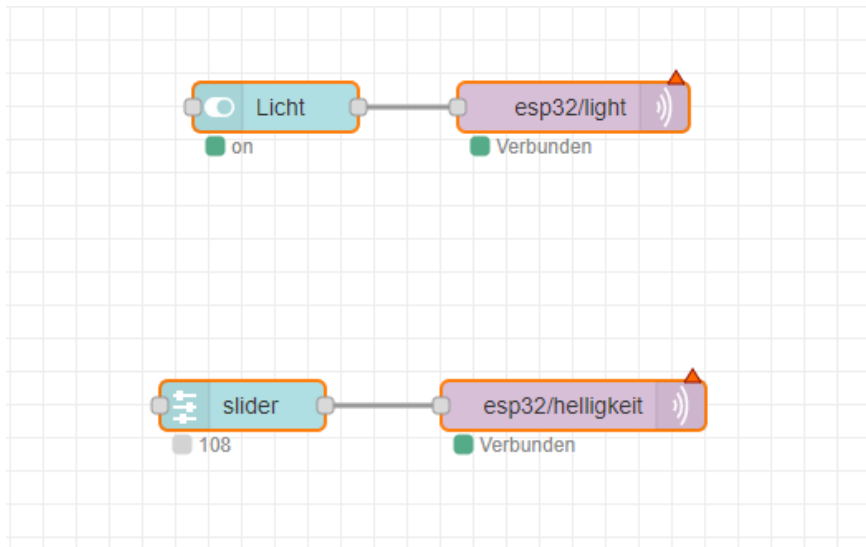
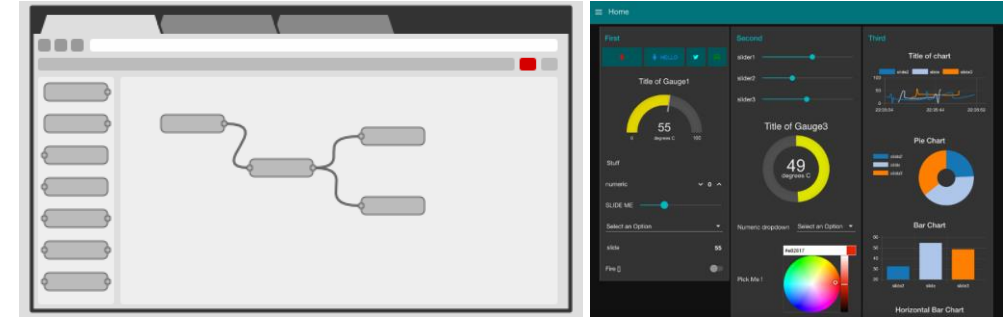
MQTT Publish / Subscribe



NodeRed

Node Red ist eine visuelle Programmierumgebung

- › Erstellung von Flows (<http://192.168.8.11:1880>)
- › Dashboard anzeigen (<http://192.168.8.11:1880/ui>)
- › Dokumentation auf <https://nodered.org/>



Am einfachsten kann MQTT durch anpassen des Beispielprogramms erfolgen

- › Daten senden in der loop() über

```
client.publish("/topic/name", "23.3")
```

- › Kanäle abonnieren in der reconnect() über

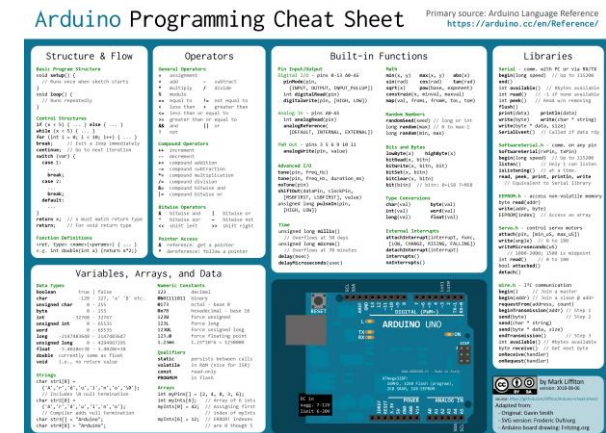
```
client.subscribe("topic/name");
```

- › Auf neue Nachrichten reagieren in der callback(...) über

```
if (String(topic) == "topic/name") {  
    if(messageTemp...)  
}
```

Wie geht es weiter?

- › **Arduino Reference** (<https://www.arduino.cc/reference/en/>)
- › **NodeRed** (<https://nodered.org/docs/user-guide/>)
- › **JavaScript** (<https://www.w3schools.com/js/default.asp>)
- › **Arduino Cheat Sheet** (Achtung Unterschiede zum NodeMCU beachten)
- › **Beispielprogramme** (<https://gitlab.sickcn.net/CD-HR/Ausbildung-IT/praktikantenprojekte/ssu-training-2021>)
- › **Betreuer fragen ;-)**



Bildnachweise

- › [https://de.wikipedia.org/wiki/Raspberry_Pi#/media/Datei:Raspberry_Pi_2_Model_B_v1.1_top_new_\(bg_cut_out\).jpg](https://de.wikipedia.org/wiki/Raspberry_Pi#/media/Datei:Raspberry_Pi_2_Model_B_v1.1_top_new_(bg_cut_out).jpg)
- › <https://joy-it.net/de/products/SBC-NodeMCU-ESP32>
- › <https://circuits4you.com/wp-content/uploads/2018/12/ESP32-Pinout.jpg>
- › <https://nodered.org/about/resources/>
- › <https://flows.nodered.org/node/node-red-dashboard>
- › https://www.informatik-aktuell.de/fileadmin/templates/wr/pics/Artikel/03_Betrieb/Netzwerk/mqtt_abb1_florian_raschbichler.png
- › <https://mosquitto.org/>
- › <https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/>