WiFi und MQTT

WiFi

```
#include <ESP8266WiFi.h>

const char* ssid = “XXXX";
const char* password = “XXX";

WiFiClient espClient;
```

Verwendet wird die Arduino Library ESP8266WiFi.h
Zur Verbindung ins Internet werden ssid und password benötigt. „zuhause“ geht das recht problemlos. Eduroam geht nicht so einfach (weil ein Benutzer mit angegeben werden muss und die Firewall höher ist).

WiFi

```
void setup() {
  …
 WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  …
}
```

Hier wird die Verbindung aufgebaut und überprüft, ob die Verbindung steht.

Beispiele zu WiFi werden in die Arduino IDE eingefügt, sobald man die Bibliothek installiert.

MQTT

Verwendet wird der Arduino MQTT Client von https://github.com/knolleary/pubsubclient/

Wenn man dieses repository installiert, erscheinen MQTT Beispiele in der IDE.

Dokumentation zum MQTT Client auf https://pubsubclient.knolleary.net/api.html

MQTT

```
#include <PubSubClient.h>

const char* mqtt_server = "broker.mqtt-dashboard.com";
PubSubClient client(espClient);
char msg[50];
char buf[10];
```

Als MQTT Broker kann z.B. der öffentliche Broker `broker.mqtt-dashboard.com` verwendet werden.

`Mit PubSubClient client(espClient);` wird der Client als Objekt “espClient” initialisiert.

Die zwei char Arrays dienen der Zwischenspeicherung von Topic und Payload.

MQTT

```
void setup() {
  …
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
 …
}
```

In setup() wird die Verbindung zum Broker an Port 1883 aufgebaut.
setCallback(…) bestimmt die Funktion, die bei einkommenden Botschaften aufgerufen wird.

MQTT

```
void callback(char* topic, byte* payload, unsigned int length) {

  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

}
```

In diesem Beispiel für die callback Routine werden topic und payload auf dem seriellen Bus ausgegeben, so dass man sie mit dem seriellen Monitor ansehen kann. Das ist nur zum Kennenlernen und zur Fehlersuche gut.

In dieser Routine reagiert der Sketch auf Botschaften. Topic und Payload müssen abgefragt werden. Je nach Inhalt wird dann reagiert.

MQTT

```
#include <stdlib.h>
char buf[10];

void loop() {
  …
  itoa(IR_LED_Intensitaet, buf, 10);

  client.publish("meterValue", buf);
  …
}
```

Das Publizieren geht sehr einfach mit der Funktion publish(topic, payload);
Hier habe ich den String für die payload aus einem Zahlenwert mit itoa(…) (Integer to ASCII) aus der Standard-Bibliothek stdlib.h generiert.

MQTT

```
void loop() {
  …
   if (!client.connected()) {
    reconnect();
  }
  client.loop();
  …
}
```

In der Schleife loop() sollten noch client.loop(); stehen. Damit wird regelmäßg nachgefragt, ob eine Botschaft angekommen ist (und gegebenenfalls die callback() Routine aufgerufen).

Wenn das Programm nur publizieren, aber nicht subskribieren soll, wird client.loop(); nicht benötigt.

`if (!client.connected())` überprüft, ob die Verbindung zum Broker noch steht und versucht sich sonst neu zu verbinden.
Für die reconnect() Funktion schauen Sie am besten im MQTT Beispiel nach. Erste Versuche gehen auch ohne reconnect().