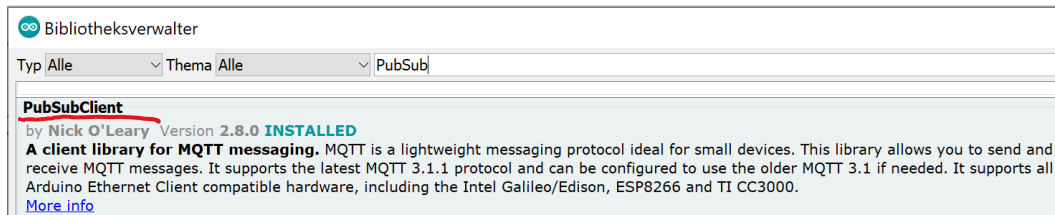
 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_vC.docx
	Datenaustausch per MQTT-Protokoll	3.1.1

Daten per MQTT verschicken

Zur Kommunikation über MQTT muss zunächst der PubSubClient über den Bibliotheksverwalter installiert werden:



Mit dem PubSubClient kann eine MQTT-Nachricht gesendet (publish) oder empfangen (subscribe) werden. Im folgenden Beispielsketch wird bei Drücken des Tasters abwechselnd die Nachricht "ON" und "OFF" geschickt. Die wichtigsten Code-Zeilen sind hier erläutert:

`#include <PubSubClient.h>`

↗ Einbinden der Bibliotheksfunktionen

`PubSubClient client(broker, 1883, espClient);` ← Neues PubSubClient-Objekt

↗ IP-Adresse des MQTT-Brokers
↗ Standard-Port für MQTT
↗ Name des Wifi-Clients


`client.connected()` ← `connected()`-Methode liefert den Verbindungsstatus (true/false)

`client.connect(mqtt_client)` ← Erstellt eine Verbindung zum Broker
↗ Jeder MQTT-Client benötigt einen eindeutigen Namen

`client.publish("schule/01/taster", "ON")` ← Nachricht an Broker schicken
↗ Topic
↗ Nachricht

Arbeitsauftrag

1. Erweitern Sie den Sketch um einen PubSub-Klienten, wie im folgenden Listing gezeigt. Ändern Sie die IP-Adresse des Brokers auf die Adresse des verwendeten MQTT-Brokers ab.
2. Starten Sie den MQTT-Client über ein Terminal (SSH-Verbindung zum Raspberry) mit dem folgenden Aufruf: `mosquitto_sub -h 192.168.4.1 -t „home/schule/#“` (Ändern Sie die IP-Adresse auf die Adresse des verwendeten MQTT-Brokers ab).

 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_vC.docx
	Datenaustausch per MQTT-Protokoll	3.1.2

Beispiel: Tastendruck als Schaltersignal per MQTT publizieren

```
#include <WiFi.h>
#include <PubSubClient.h>
#define TasterPin 19
#define PRESS 0

const char* ssid = "piNet3";           // Name des WLAN-Netzes
const char* password = "lfb_1234";     // Passwort des WLAN-Netzes
const char* mqtt_client = "Schule01";  // hier eindeutigen Namen vergeben!!
const char* broker = "broker.hivemq.com"; // IP-Adresse des MQTT-Brokers im Netz


WiFiClient espClient;                  // WiFi-Client Objekt
PubSubClient client(broker, 1883, espClient); // PubSub-Client Objekt

void wifiConnect() { ... }

void mqttConnect()
{
  while (!client.connected())
  {
    Serial.println(); Serial.println("Connect to Broker..");
    if (client.connect(mqtt_client)) // Starte MQTT-Verbindung...
      Serial.println(mqtt_client);
  }
}

void setup()
{
  pinMode(TasterPin, INPUT);
  Serial.begin(9600);
  wifiConnect();
}

uint32_t oldTime;
bool toggleState;
void loop()
{
  if (!client.connected()) mqttConnect();
  uint32_t newTime = millis();
  if(digitalRead(TasterPin) == PRESS)
  {
    if (newTime - oldTime > 50)
    {
      toggleState = !toggleState;
      if (toggleState == 1) client.publish("schule/01/taster", "ON");
      else client.publish("schule/01/taster", "OFF");
    }
    oldTime = newTime;
  }
}
```

 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_VC.docx
	Datenaustausch per MQTT-Protokoll	3.1.3

1.1 DHT-Sensordaten per MQTT publizieren

Mit dem PubSub-Client sollen nun die Temperatur und Luftfeuchte-Werte verschickt werden.


Arbeitsauftrag

1. Ändern Sie den letzten Sketch so ab, dass wieder die dht-Bibliothek eingebunden wird und ein Sensor-Objekt erstellt wird.
2. Speichern Sie in der loop()-Funktion die Sensordaten wieder zyklisch (5s) in den beiden float-Variablen **cTemp** und **hum**.
3. Da die client.publish()-Methode Arrays vom Typ char verlangt, müssen die float-Werte vor dem versenden in einen Zeichenketten-Puffer geschrieben werden. Dies erledigt die C-Funktion sprintf(), die auch gleich eine einfache Formatierung der Zeichenkette auf 4 Stellen mit einer Nachkommastelle erlaubt. Bauen Sie den angegebenen Code in der loop()-Funktion ein.

```
char messageBuffer[10];
sprintf(messageBuffer, "%4.1f°C", cTemp);
client.publish("home/schule/1/cTemp", messageBuffer);

sprintf(messageBuffer, "%4.1f%", hum);
client.publish("home/schule/1/hum", messageBuffer);
```

4. Übertragen Sie den kompletten Sketch auf die Zielhardware.
5. Testen Sie den MQTT-Empfang wieder mit mosquitto_sub in einer ssh-Sitzung mit puTTY.

 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_vC.docx
	Datenaustausch per MQTT-Protokoll	3.1.4

Lösung: Vollständiger Programmcode ohne wifiConnect() und mqttConnect().

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

const char* ssid = "piNet3";           // Anmeldedaten für's WLAN
const char* password = "lfb_1234";
const char* mqtt_client = "schule01";  // hier eindeutigen Namen vergeben!!
const char* broker = "broker.hivemq.com"; // Adresse des MQTT-Brokers im Netz

#define DHTPIN 5
#define DHTTYPE DHT11

WiFiClient espClient;                  // WiFi-Client Objekt
PubSubClient client(broker, 1883, espClient); // PubSub-Client Objekt
DHT dht(DHTPIN,DHTTYPE);              // DHT-Sensor Objekt

void wifiConnect() {...}

void mqttConnect() {...}

void setup()
{
  Serial.begin(9600);
  dht.begin();
  wifiConnect();
}


void loop()
{
  if (!client.connected()) mqttConnect();

  float hum    = dht.readHumidity();
  float cTemp  = dht.readTemperature();

  char messageBuffer[10];
  sprintf(messageBuffer,"%4.1f°C", cTemp);
  client.publish("schule/01/cTemp", messageBuffer);

  sprintf(messageBuffer,"%4.1f%%", hum);
  client.publish("schule/01/hum", messageBuffer);

  delay(5000);           // Messwerte alle 5s verschicken
}
```

 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_vC.docx
	Datenaustausch per MQTT-Protokoll	3.1.5

1.2 Sensorwerte als JSON-Container verschicken

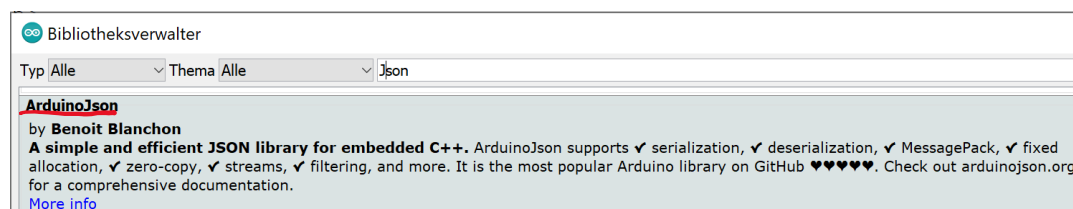
Die JavaScript-Object-Notation (JSON) bietet sich an, wenn mehrere Daten gleichzeitig verschickt oder empfangen werden sollen. Innerhalb eines JSON-Objekts werden die Daten mittels Schlüssel/Werte-Paaren repräsentiert. Für die Datenübertragung muss das Objekt serialisiert werden. Dies bedeutet, dass es in eine flache Zeichenkette umgewandelt werden muss. Diesen Vorgang nennt man **Serialisieren**. Für die DHT11-Sensorwerte würde das dann wie folgt aussehen:

```
{
  "cTemp": 24,
  "hum": 49
}
```

Serialisieren →

```
"{"cTemp":24,"hum":49}"
```

Für das Handling von JSON-Objekten in Arduino-Sketchen ist es sinnvoll, einen JSON-Parser zu installieren.



Arbeitsauftrag

1. Binden Sie den JSON-Parser in den Sketch ein
`#include <ArduinoJson.h>`
2. Ändern Sie die `loop()`-Funktion folgendermassen ab:

```
void loop()
{
  if (!client.connected()) mqttConnect();

  StaticJsonDocument<256> jsonDoc;           // Neues statisches JSON-Doc anlegen


  jsonDoc["cTemp"] = dht.readTemperature(); // Sensorwerte in das JSON-Doc eintragen
  jsonDoc["hum"]   = dht.readHumidity();

  String serialized = "";
  serializeJson(jsonDoc, serialized);        // Serialisieren des JSON-Doc

  char messageBuffer[serialized.length()+1]; // Zeichenketten-Array anlegen
  serialized.toCharArray(messageBuffer, serialized.length()+1); // String in Array kopieren

  client.publish("schule/01/sensor", messageBuffer); // und Publizieren
  delay(5000);
}
```

3. Testen Sie wieder mit `mosquitto_sub`.

 Friedrich-Ebert-Schule Esslingen FES	IT: Internet Of Things	Name: Rahm Datum: 08.02.2023 3_2_MQTT_auf_vC.docx
	Datenaustausch per MQTT-Protokoll	3.1.7

- loop()-Funktion

In der loop()-Funktion muss zyklisch die client.loop()-Methode aufgerufen werden. Ansonsten kann die loop() hier weitestgehend leer bleiben.

```
void loop()
{
  if (!client.connected()) mqttConnect();
  client.loop();
  delay(20);
}
```

Testen Sie das Programm, indem Sie einen Schaltbefehl auf das Topic publizieren:

```
>mosquitto_pub -h 192.168.4.1 -t "schule/01/led" -m "ON"
```

und

```
>mosquitto_pub -h 192.168.4.1 -t "schule/01/led" -m "OFF"
```