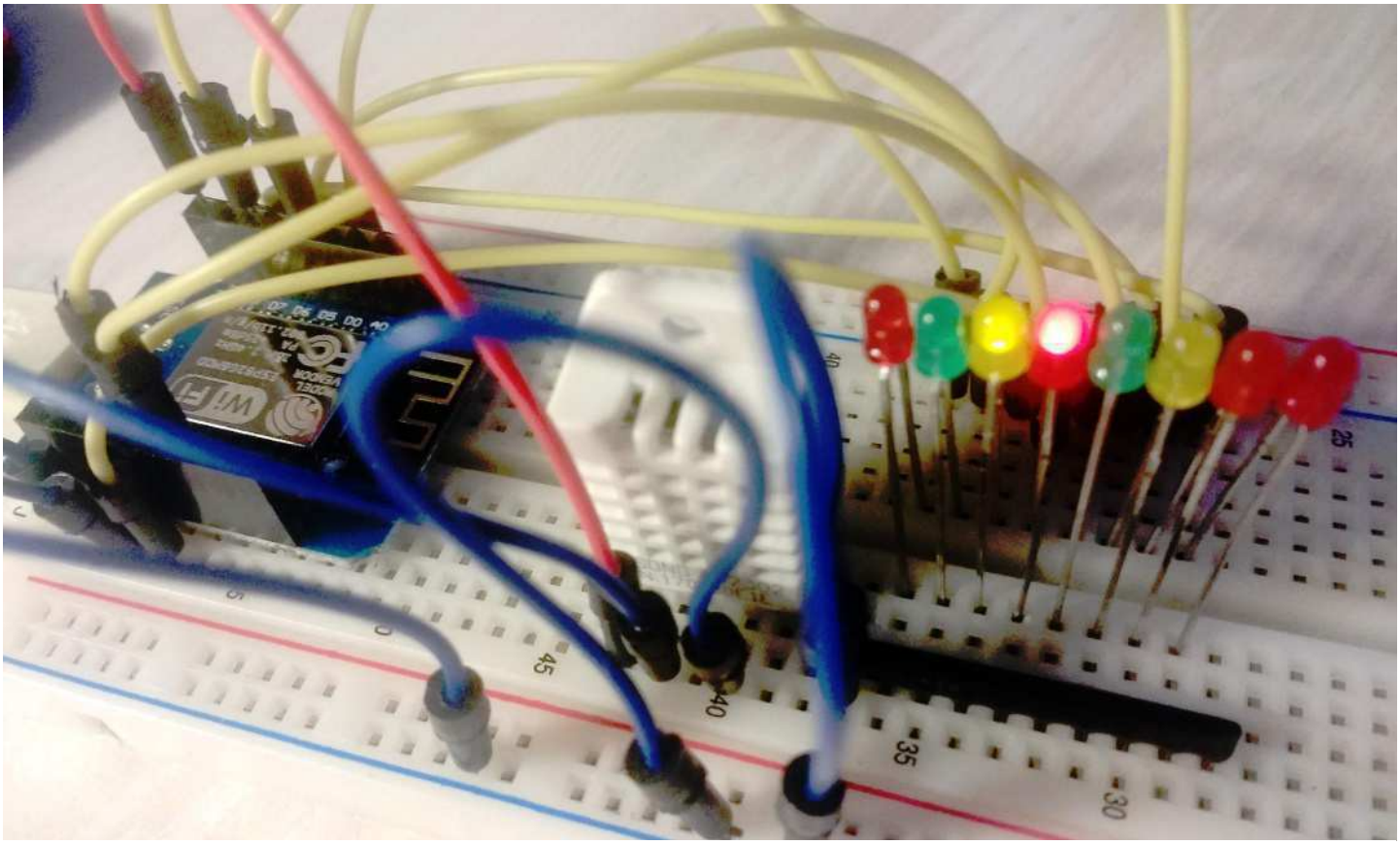
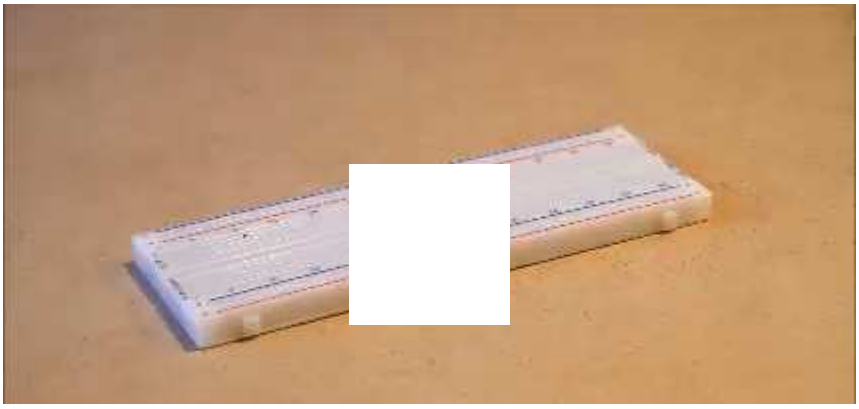


IoT – An introduction to the Internet Of Things



00:00 00:36

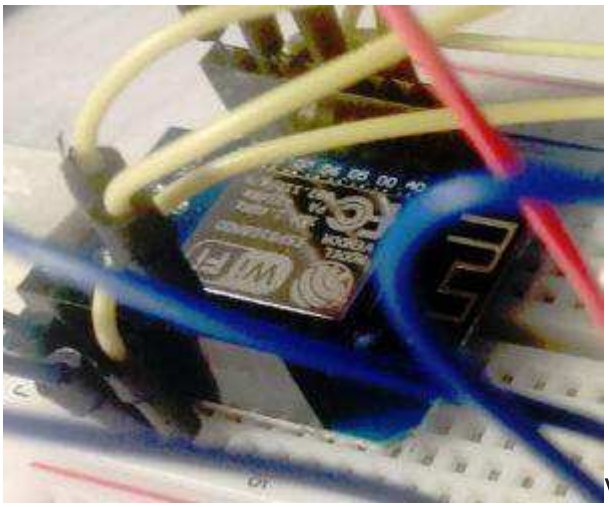
In the future, it can be expected that all electronic devices are connected to the Internet. This makes it possible, for example, to view information from these devices via the smartphone anywhere in the world or to control the devices remotely.

Especially successful at the moment are products like Amazon Alexa, Philipps Hue or Ikea TRÅDFRI who work on the basis of these technologies.

As part of this course, you will learn the basics of networking devices in the Internet of Things to understand the basic principles, opportunities and dangers of these technologies.

You plan, program and build your own devices for the world of the Internet of Things.

Prepare for IoT

 Very suitable for first steps is the extremely inexpensive microcontroller WMOS D1 Mini with integrated WIFI chip of the type ESP-8266EX.

This course will develop a device to record the illuminance, temperature and humidity of a room. Later the control of a fan and a heater should be possible.

information

Prerequisites in knowledge

In order to be able to work on this course, it is helpful to have basic knowledge in the programming of Arduino microcontroller platforms.

This course can be mastered without any problems if the following courses have been previously taken:

- Introduction to Processing (basics of programming)
- Introduction to Arduino (basics of Arduino plattform)

Prerequisites in materials

To start this course the following materials need to be provided.

No	Item	Amount	Note
1.	WMOS D1 Mini	1	microcontroller board which has to be programmed for implementin the control logic and connection to the cloud
2.	LDR	1	Simple analog sensor, to measure illuminance
3.	DHT11 (or DHT22)	1	Digital and more intelligent sensor, to measure temperature and moisture
4.	LED	1	Signal LEDs to indicate some status
5.	Resistance	1	Resistance to adjust the LED and transistor current
6.	Breadboard	1	Breadboard to assemble the circuit

information

Commissioning of the WEMOS D1 Mini

First approach to realize how a microcontroller plattform works is to implement really simple solutions to understand the process of connecting hardware to the board and develop appropriate programs.

Working task

> Commission the microcontroller using the manufacturer's instructions, program the device with the example code shown in the information section and connect LEDs to display the output signals.

information

Programming of the microcontroller

For programming the Arduino development environment can be used. On the manufacturer's website of the microcontroller is a good guide available that shows how the microcontroller can be programmed.

Use of the IOs

The inputs and outputs of the microcontroller can be used in the usual Arduino way.

Below is an example to be developed in the context of this project (so it may seem more complicated than necessary at the moment).

```
#include <ESP8266WiFi.h>

// local hardware
const int leds[]={D7, D8};
const int ledCount = 2;
const int signalFan = D7;
const int signalHeating = D8;

int blink = LOW;

void setup() {
 Serial.begin(9600);
```

```
 for(int i=0; i<ledCount; i++)
   pinMode(leds[i], OUTPUT);
}

void processOutputs() {
 digitalWrite(signalFan, blink);
 digitalWrite(signalHeating, blink);
}

void loop() {
 blink = blink==LOW ? HIGH : LOW;
 processOutputs();
 delay(1000);
}
```

information

Connection of a sensor

In the next step, the sensor for the detection of humidity and temperature should be added.

For this project, the popular digital sensor DHT11 or DHT22 (more expensive but more accurate) is used.

Working Task

Commission the sensor and the additional signal lamps in the following steps:

- Install the required library for the DHT22 sensor
- Create a functional description of the program shown in the information section
- Connect the LEDs as signal lights to the outputs specified by the program

information

Below is a program code that has been supplemented with the integration of the DHT22 sensor and other signal lights for the display of temperature and humidity limits.

```
#include <ESP8266WiFi.h>
#include <DHT.h>

#define DHTPIN D4
#define DHTTYPE DHT22 // DHT 22 (AM2302)

// Sensor
DHT dht(DHTPIN, DHTTYPE);
long lastSensorTime = 0;
float lastTemperature = 0.0f;
float lastHumidity = 0.0f;

// local hardware
const int leds[]={D0, D1, D2, D3, D5, D6, D7, D8};
const int ledCount = 8;
```

```
const int signalTCold = D0;
const int signalTOK = D1;
const int signalTHot = D2;
const int signalHDry = D3;
const int signalHOK = D5;
const int signalHWet = D6;
const int signalFan = D7;
const int signalHeating = D8;
bool activateFan=false;
bool activateHeating=false;

void processInputs() {
 long now = millis();
 float t= dht.readTemperature();
 float h= dht.readHumidity();
 if(!isnan(t) && !isnan(h)) {
   lastSensorTime = now;
   lastTemperature = t;
   lastHumidity = h;
 }
}

void processOutputs() {
 digitalWrite(signalTCold, LOW);
 digitalWrite(signalTOK, LOW);
 digitalWrite(signalTHot, LOW);
 if(lastTemperature < 18){
   digitalWrite(signalTCold, HIGH);
 } else if(lastTemperature > 22) {
   digitalWrite(signalTHot, HIGH);
 } else {
   digitalWrite(signalTOK, HIGH);
 }

 digitalWrite(signalHWet, LOW);
 digitalWrite(signalHOK, LOW);
 digitalWrite(signalHDry, LOW);

 if(lastHumidity < 50){
   digitalWrite(signalHDry, HIGH);
 } else if(lastHumidity > 60) {
   digitalWrite(signalHWet, HIGH);
 } else {
   digitalWrite(signalHOK, HIGH);
```

```
 }
 digitalWrite(signalFan, activateFan ? HIGH : LOW);
 digitalWrite(signalHeating, activateHeating ? HIGH : LOW);
}

void loop() {
 processInputs();
 processOutputs();
}
```

information

Establish Wifi-connection

For the real IoT application, the microcontroller still needs to be able to spark into the network. For this purpose, a WIFI connection is established in the following step.

working task

Establish a connection by incorporating the code listed in the information section into the existing program with the WIFI router and adjust the source code so that the signal LEDs indicate the search as running light during connection setup.

information

Wifi connection

The following code enables the connection to a WIFI access point or router.

```
#include <ESP8266WiFi.h>
#include <DHT.h>

// WIFI Connection
const char* ssid = "Fritzbox1";
const char* password = "Fritzbox1Passwort";

WiFiClient espClient;

// local hardware
[...]

void setup() {
 [...]
 setupWifi();
}

void setupWifi() {
 delay(100);
 Serial.print("Connecting to ");
 Serial.print(ssid);
 Serial.print(" ");
```

```
  Serial.print("   ");

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  randomSeed(micros());
  Serial.println("");
  Serial.print("WiFi connected - ");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

} // setupWIFI

[...]
```

information

Publish sensor data

After an existing internet connection, the data should now be sent to the cloud. For this the protocol MQTT is used.

working task 1

Apply the above additions to the source code in your program. After successful transfer to the microcontroller, the sensor data should be sent to the broker.

working task 2

Review your understanding of the illustrated program by answering the following questions:

- How often are the sensor data sent to the broker?
- Which topics were chosen for the sensor data?

working task 3

Install an MQTT app (such as MQTT Dash) to display the published data.

information

MQTT

With the help of the MQTT protocol z. B. data to a so-called “broker” (= a server on the Internet, which controls the MQTT protocol) are sent.For initial tests, brokers that are freely accessible on the Internet can be used. It should be noted, however, that the data sent there can be viewed and manipulated by anyone. Therefore, no security-related or personal data should be sent.

The data is sent to the server under unique names, the so-called “Topics”. The topics are noted as strings with slashed sections to allow mapping of hierarchies. If, for example, the temperatures of each room of a house are recorded, the following description for the topics would be conceivable:

- /MyHouse/Kitchen/Temperature
- /MyHouse/LivingRoom/Temperature
- ....

Once the data has been sent to the server, they can be viewed using an app that retrieves the data from the broker via the MQTT protocol.

MQTT sample code

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// MQTT Connection
String HOST_NAME="Klasse.Schüler";
const char* mqttServer = "iot.eclipse.org";
PubSubClient mqttClient(espClient);
long lastMessageTime = 0;

[...]

void setup() {
 [...]
 setupMQTT();
}

[...]

void setupMQTT() {
 mqttClient.setServer(mqttServer, 1883);
}

void connectMQTT() {
 // Loop until connection is established
 while (!mqttClient.connected())
 {
   Serial.print("Try to connect MQTT ...");
   // random or unique mqttClient ID
   String mqttClientId = String("CRBK_")+String(HOST_NAME);

   // for authentification use username and password like: mqttClient
   if (mqttClient.connect(mqttClientId.c_str()))
   {
      Serial.println("connected");
   } else {
     Serial.print("failed, errorcode=");
```

```
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
} //end connectMQTT()

void processMQTT() {
 if (!mqttClient.connected()) {
   connectMQTT();
 }
 mqttClient.loop();
 long now = millis();
 if (now - lastMessageTime > 6000 && lastSensorTime!=lastMessageTime)
   lastMessageTime = lastSensorTime;

   //publish sensor data to MQTT broker
   Serial.println("publish on server");
   mqttClient.publish(("CRBK/"+HOST_NAME+"/Temperature").c_str(), Str
   mqttClient.publish(("CRBK/"+HOST_NAME+"/Humidity").c_str(), String
 }
}

void loop() {
 processInputs();
 processMQTT();
 processOutputs();
}
```

information

Receiving control commands

In the last chapter, you learned how to send data to the broker.

In the next step, the IoT device should be able to receive data in order to be remotely controlled.

working task 1

Apply the changes to the reaction to external control commands. Use an MQTT app to control the fan.

working task 2

Complete the program to control the heating with the variable “ActivateHeating”.

information

MQTT Callback

For this purpose, a “callback” function must be added. This callback function is automatically called by the system as soon as a broker issues a control command. In order for the IoT device to receive the commands, the device must log in to the broker for the notification (subscribe).

```
[...]

void setupMQTT() {
 mqttClient.setServer(mqttServer, 1883);
 mqttClient.setCallback(mqttCallback);
}

void mqttCallback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Command is: ");
 Serial.println(topic);
 int p = (char)payload[0] - '0';
 if(("CRBK/"+HOST_NAME+"/ActivateFan")==String(topic)) {
   activateFan = p!=0;
 }
```

```
 Serial.println();
} //end callback

void connectMQTT() {
 // Loop until we're reconnected
 while (!mqttClient.connected())
 {
   Serial.print("Try to connect MQTT ...");
   // random/unique mqttClient ID
   String mqttClientId = String("CRBK_")+String(HOST_NAME);

   // for authentification use username and password like: mqttClient
   if (mqttClient.connect(mqttClientId.c_str()))
   {
     Serial.println("connected");
     //once connected to MQTT broker, subscribe command
     mqttClient.subscribe(("CRBK/"+HOST_NAME+"/ActivateFan").c_str())
   } else {
     Serial.print("failed, errorcode=");
     Serial.print(mqttClient.state());
     Serial.println(" try again in 5 seconds");
     delay(5000);
   }
 }
} //end connectMQTT()

[...]
```

information

OTA – Over The Air

In order to enable the programming of IoT devices even without a line connection, the WMOS microcontroller can also be programmed with little effort over the air (WIFI).

working task

Integrate the code shown in the information section into your own program to implement the OTA feature

information

OTA sample code

In order to implement the OTA programming, the program must be supplemented once with code which enables programming via the WIFI connection during operation.

information

```
#include <ESP8266WiFi.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#include <DHT.h>

[...]

void setupWifi() {
  [...]
  // default port is 8266
  // ArduinoOTA.setPort(8266);

  //default hostname is esp8266-[ChipID]
  ArduinoOTA.setHostname(HOST_NAME.c_str());

  // add password for authentication
  // ArduinoOTA.setPassword((const char *)"123");

  ArduinoOTA.onStart([]() {
    Serial.println("Start");
```

```
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("End");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total)
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed")
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Fail
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Fail
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin();
} // initWIFI

void loop() {
 ArduinoOTA.handle();
 [...]
}
```

After programming the microcontroller via the USB line with the above program, the Arduino development environment will automatically provide a network programming port.

The following programming can be done already OTA.