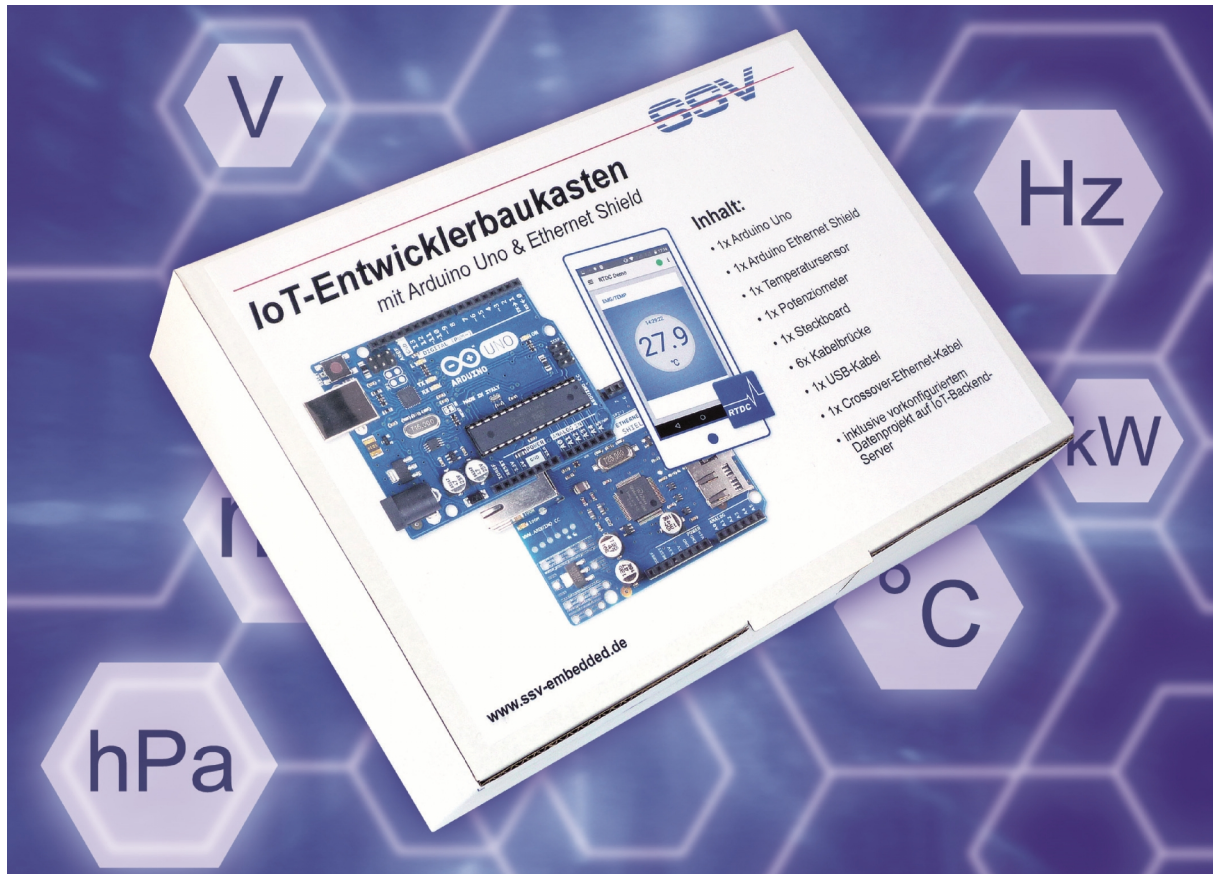


## IoT-Entwicklerbaukasten – Erste Schritte

Der IoT-Entwicklerbaukasten dient in erster Line dazu, in möglichst kurzer Zeit einen Feld-daten-Frontend zu schaffen, um „echte“ Sensordaten an eine Cloud- bzw. IoT-Service-plattform im Internet zu schicken. Anschließend können die Daten über eine PC-Webseite bzw. Smart-App in Echtzeit visualisiert werden.



In dieser kurzen Beschreibung finden Sie alle Informationen, um innerhalb von wenigen Minuten per *Arduino Uno* die Temperatur in Ihrer Umgebung ins Internet zu übertragen und mit einer Webseite zu betrachten.

### Wichtige Voraussetzungen

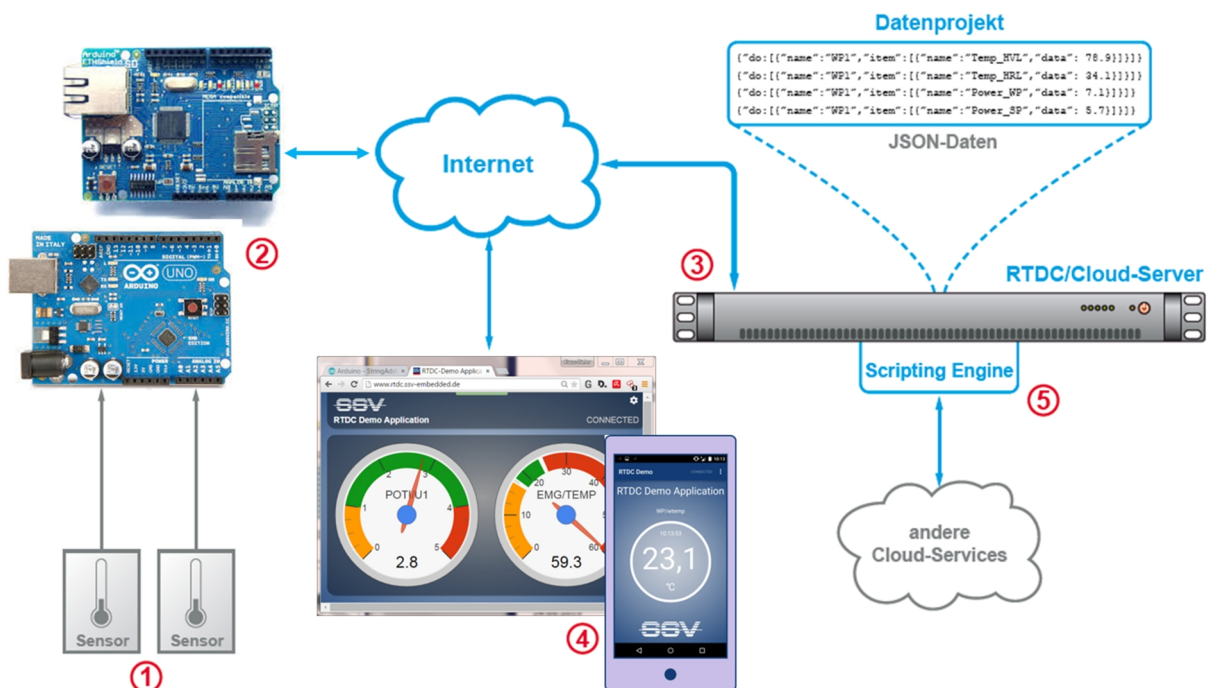
Wir gehen davon aus, dass Sie bereits erste Erfahrungen mit Arduino-Baugruppen gesammelt und auch die Arduino-Entwicklungsumgebung (Arduino-IDE) mit einem USB-Treiber für Arduino Uno auf Ihrem PC installiert haben. Ansonsten sollten Sie sich zunächst unter <https://www.arduino.cc/> mit der entsprechenden Thematik vertraut machen.

## Lieferumfang und Zielsetzung

Die folgende Tabelle liefert Ihnen einen Überblick zum Lieferumfang Ihres IoT-Entwicklerkits.

Bauteil	Funktion	Anz.
<a href="#">Arduino Uno</a>	Embedded System	1x
<a href="#">Arduino Ethernet Shield</a>	LAN Interface	1x
LM35CZ TO-92	Temperatursensor	1x
P06M-LIN 10K	Potenzimeter	1x
Steckboard-Erweiterung	Steckboard	1x
Kabelbrücken Blau	Kabel für Sensor	2x
Kabelbrücken Rot	Kabel für Sensor	2x
Kabelbrücken Gelb	Kabel für Sensor	2x
USB-Kabel	USB zum PC	1x
LAN-Kabel Cross-over	LAN zum PC/Router	1x

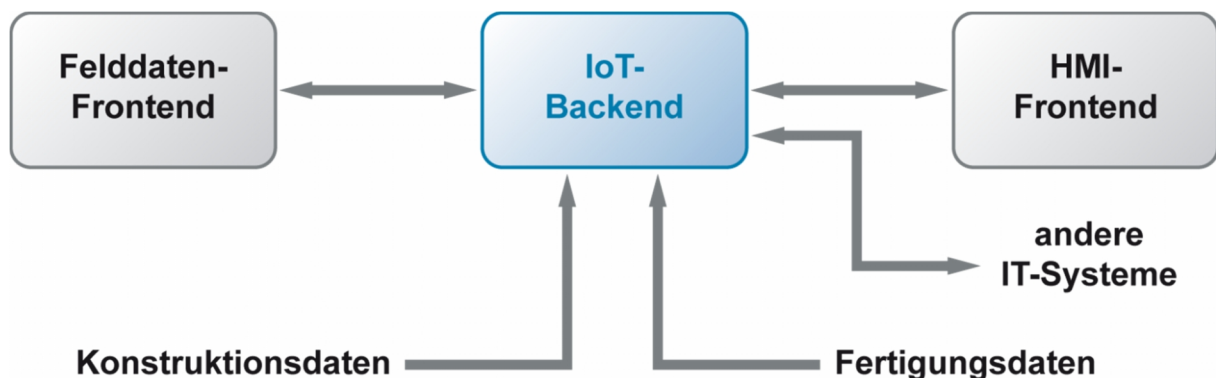
Mit diesem Lieferumfang wollen wir, wie in der folgenden Abbildung ersichtlich, eine IoT-Anwendung entwickeln, die zwei Sensordatenpunkte (1 x Spannung, 1 x Temperatur) in der Cloud abbildet.



- 1 Zwei „Sensoren“ sind über analoge (Arduino Uno) Eingänge mit einem IoT-Gateway verbunden.
- 2 Als IoT-Gateway dient ein Arduino Uno mit Ethernet Shield. Hier werden die Sensordaten digitalisiert und bei jeder Änderung (Datenvorverarbeitung) an einen IoT-Server im Internet übertragen. Sensoren und Gateway bilden zusammen den **Felddaten-Frontend**.
- 3 Auf dem Server (**IoT-Backend**) werden die Sensordaten in einem JSON-basierten Datenobjekt gespeichert, das somit die virtuelle (Daten-) Repräsentanz der Sensoren darstellt.
- 4 Ein Smartphone/Tablet mit geeigneter App oder ein Webbrowser mit einer entsprechenden Webseite kann als **HMI-Frontend** auf die virtuelle Repräsentanz der einzelnen Sensoren zugreifen und die Daten visualisieren.
- 5 Über eine Scripting Engine auf dem **IoT-Backend** können die Daten auch anderen Cloud-Servern bzw. Cloud-Services oder MES-/ERP-Anwendungen zur Verfügung gestellt werden.

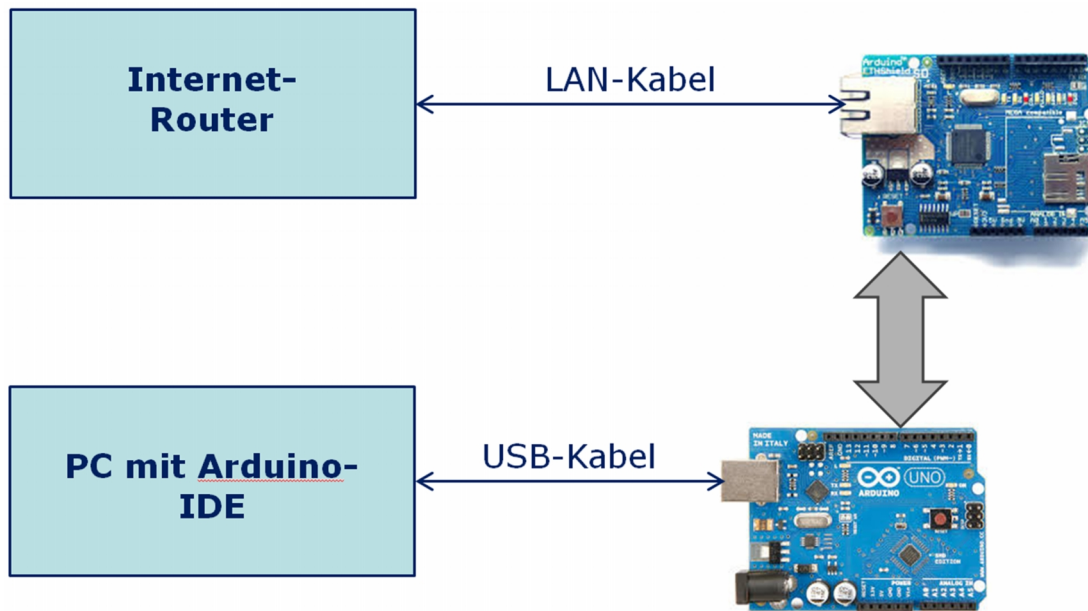
### Hinweis zu den Bausteinen einer IoT-Anwendung

SSV geht für IoT-Anwendungen grundsätzlich von den drei Bausteinen 1. **Felddaten-Frontend**, 2. **IoT-Backend** und 3. **HMI-Frontend** aus.



## Arduino vorbereiten

Verbinden Sie den Arduino Uno mit dem Ethernet Shield. Stellen Sie zwischen Ethernet Shield und Ihrem Internet-Router eine LAN-Kabelverbindung her.



Verbinden Sie den Arduino Uno per USB-Kabel mit Ihrem PC. Sorgen Sie dafür, dass auf Ihrem PC die Arduino-IDE läuft.



```

sketch_analog-bsp11 | Arduino 1.0.6
Datei Bearbeiten Sketch Tools Hilfe

sketch_analog-bsp11
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int value = analogRead(0);
  int percent = map(value, 0, 1023, 0, 100);
  Serial.print(value);
  Serial.print(" ");
  Serial.print(percent);
  Serial.print("% ");

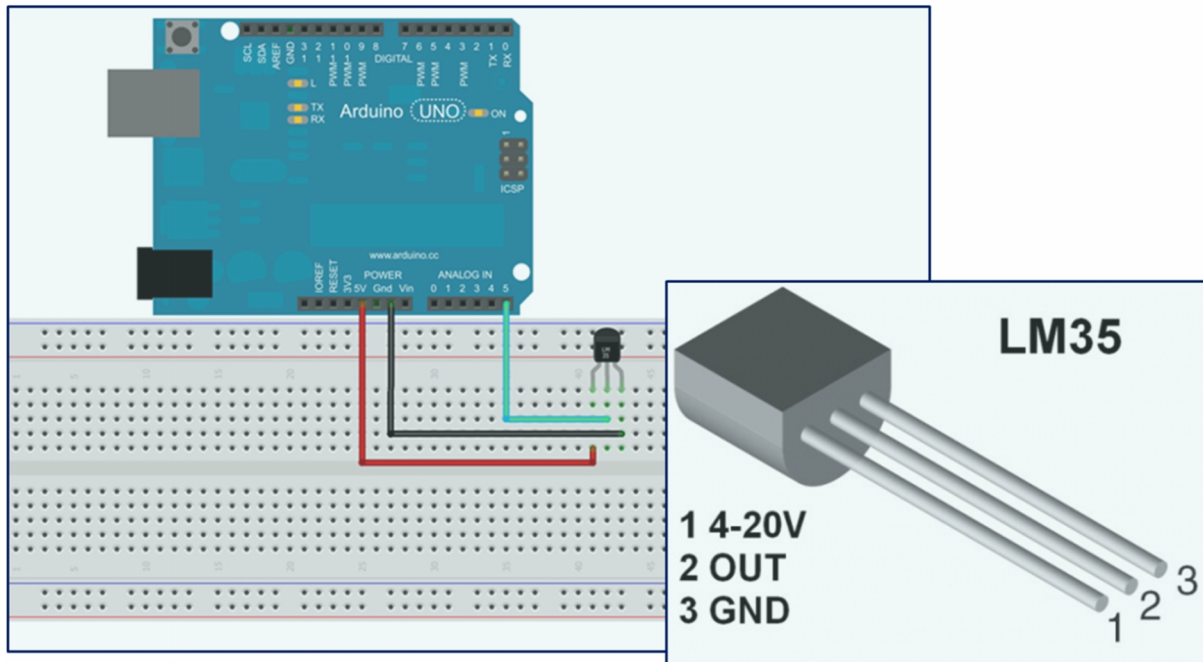
  float voltage = value * (5.0 / 1023.0);
  Serial.print(voltage);
  Serial.println("V");
  delay(1000);
}

20 Arduino Nano w/ ATmega328 on COM9
  
```



## Temperatursensor mit dem Arduino verbinden

Verbinden Sie den LM35-Temperaturseonsor wie in der folgenden Abbildung mit einem analogen Eingang des Arduino Uno.



Testen Sie Ihre Schaltung mit dem hier folgenden Code-Beispiel. Übertragen Sie den Code in die Arduino-IDE. Übersetzen Sie das Beispiel und laden das Ergebnis in den Arduino Uno. Führen Sie den Code auf dem Arduino aus und beobachten Sie die Temperatursausgabe.

```
// LM35 an A0 einlesen ...

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

void loop() {
  float raw = analogRead(0);
  float percent = raw / 1023.0;
  float volts = percent * 5.0;
  float temp = volts * 100.0;
  Serial.println(temp);
  delay(1000);
}
```

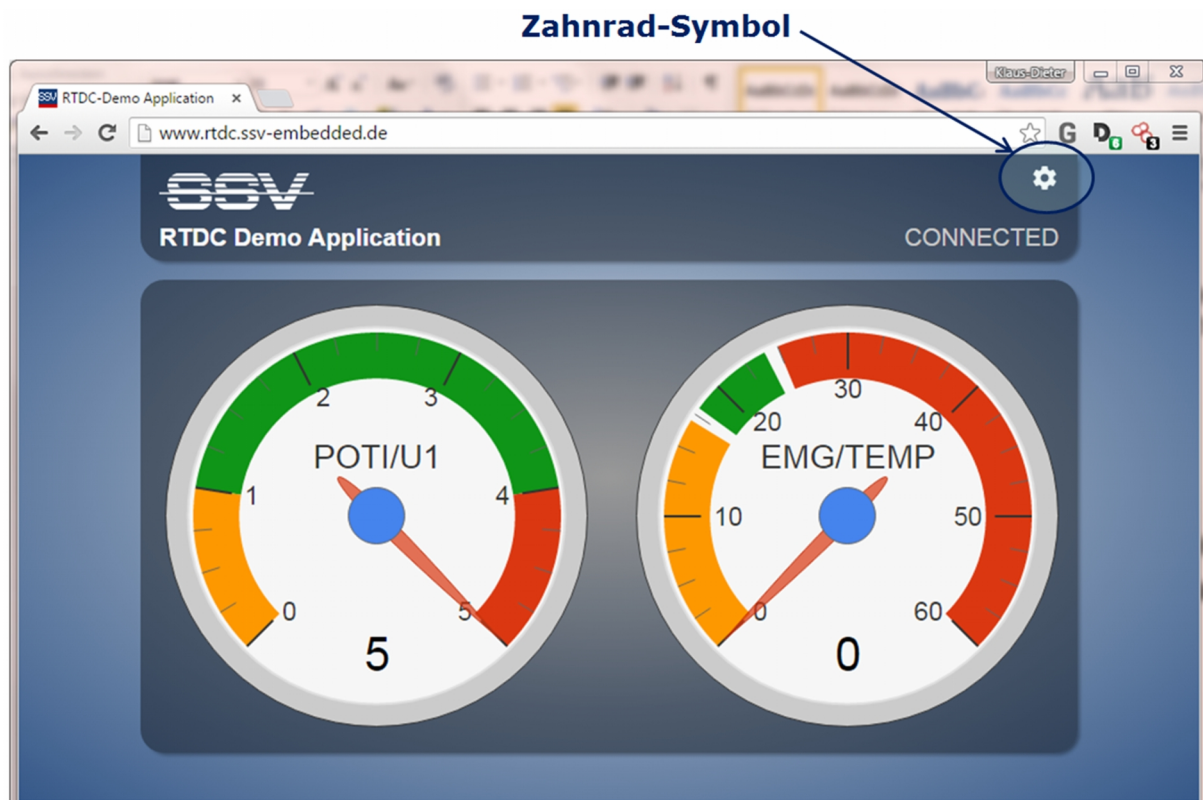
Beachten Sie bitte, dass der benutzte analoge Eingang am Arduino Uno (A0 bis A5) und im Code-Beispiel korrespondieren müssen. In der Abbildung wird A5, im Code A0 benutzt.

## Webzugriff auf die Temperaturmesswerte vorbereiten

Zum Lieferumfang Ihres IoT-Entwicklerkits gehört eine JSON-Datei mit dem Namen *IoT-Workshop\_n.json* (*n* ist eine Zahl größer 0). Bis auf die beiden Elemente **auth\_key** und **access\_key** hat diese Datei den folgenden Inhalt:

```
{
  "version": 1,
  "mqtt": "ngra-ssv.dynalias.net",
  "mqtt_port": 5083,
  "mqttts_port": 5084,
  "rest": "ngra-ssv.dynalias.net",
  "rest_port": 5080,
  "rests_port": 5081,
  "auth_key": "8f610167-4711-4698-1204-ce1e10998405",
  "access_key": "b3930eac-4411-4702-9288-8c60fc649812",
  "mqtt_time_out": 60,
  "mqtt_keep_alive": 120,
  "rest_time_out": 60,
  "ssl": false
}
```

Rufen Sie nun mit Ihrem Webbrowser (bitte eine aktuelle Firefox- oder Chrome-Version benutzen, der Internet Explorer weist etliche Inkompatibilitäten hinsichtlich HTML5 auf) den Link <http://www.rtdc.ssv-embedded.de/> auf.



Klicken Sie im Browser-Fenster oben rechts auf das Zahnrad-Symbol. Es öffnet sich ein Texteingabefenster. Übertragen Sie den Inhalt Ihrer Datei *IoT-Workshop\_n.json* (zum Beispiel per Cut & Paste [Kopieren und Einfügen]) in dieses Fenster.



Danach muss im oberen rechten Bereich des Browser-Fensters (direkt unter dem Zahnrad-Symbol) die Statusmeldung von **DISCONNECT** auf **CONNECT** umschalten.

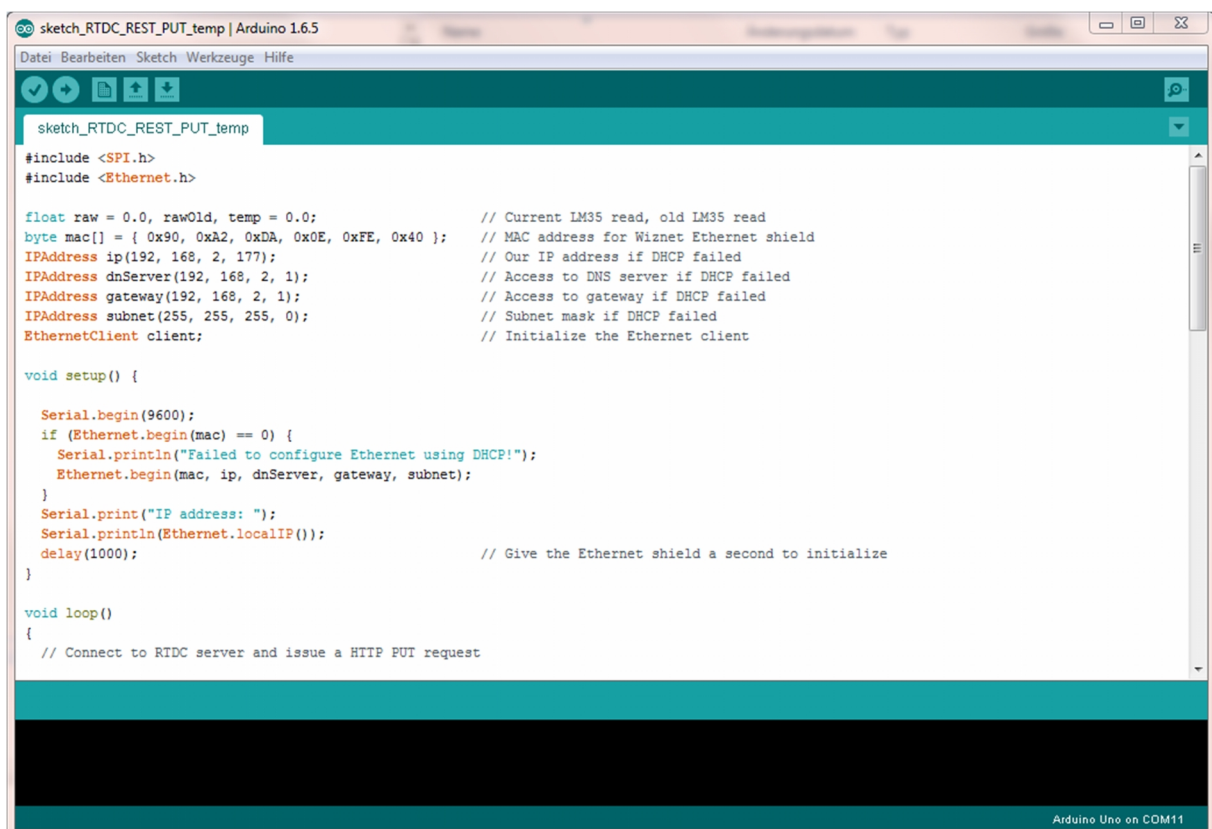
Durch einen erneuten Klick auf das Zahnrad-Symbol können Sie das Texteingabefenster wieder schließen und bei Bedarf erneut öffnen.

## Temperatur per Arduino in die Cloud senden

Erzeugen Sie in der Arduino-Anwendungsumgebung auf Ihrem PC ein neues Verzeichnis mit dem Namen *sketch\_RTDC\_REST\_PUT\_temp*.

Kopieren Sie die Datei *sketch\_RTDC\_REST\_PUT\_temp.ino* aus dem Lieferumfang Ihres IoT-Entwicklerkits in dieses Verzeichnis.

Öffnen Sie die Datei *sketch\_RTDC\_REST\_PUT\_temp.ino* dann mit Ihrer Arduino-IDE auf dem PC.



```
sketch_RTDC_REST_PUT_temp | Arduino 1.6.5
Datei Bearbeiten Sketch Werkzeuge Hilfe

sketch_RTDC_REST_PUT_temp

#include <SPI.h>
#include <Ethernet.h>

float raw = 0.0, rawOld, temp = 0.0;           // Current LM35 read, old LM35 read
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xFE, 0x40 }; // MAC address for Wiznet Ethernet shield
IPAddress ip(192, 168, 2, 177);                // Our IP address if DHCP failed
IPAddress dnServer(192, 168, 2, 1);             // Access to DNS server if DHCP failed
IPAddress gateway(192, 168, 2, 1);             // Access to gateway if DHCP failed
IPAddress subnet(255, 255, 255, 0);            // Subnet mask if DHCP failed
EthernetClient client;                        // Initialize the Ethernet client

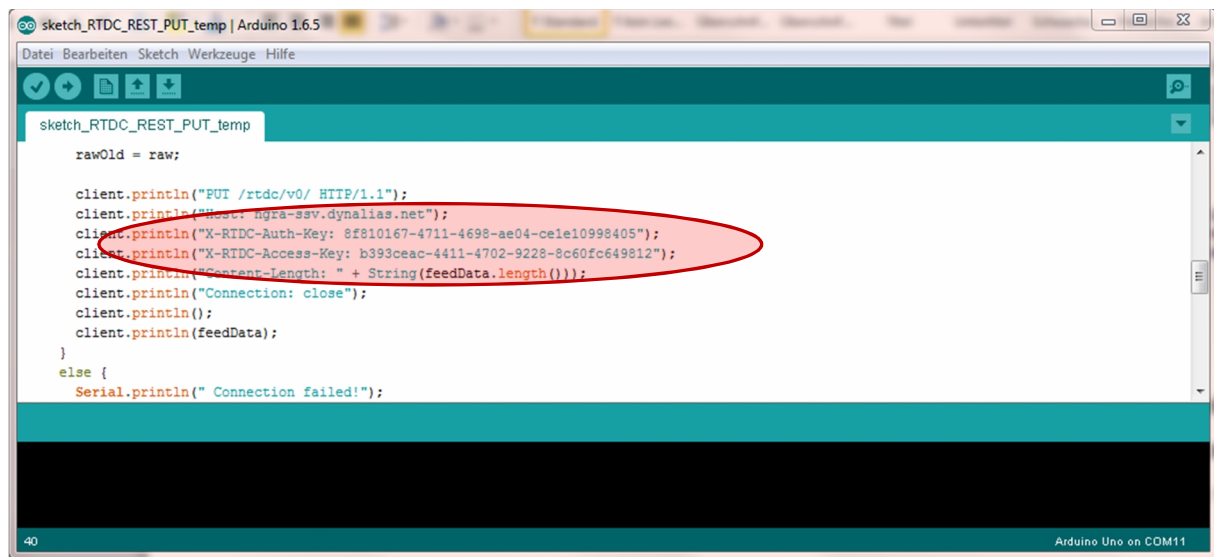
void setup() {
  Serial.begin(9600);
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP!");
    Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  }
  Serial.print("IP address: ");
  Serial.println(Ethernet.localIP());
  delay(1000);                                // Give the Ethernet shield a second to initialize
}

void loop() {
  // Connect to RTDC server and issue a HTTP PUT request
```

In den Codezeilen 40 und 41 sind die beiden Elemente **auth\_key** und **access\_key** gespeichert. Sie dienen als individuelle Zugriffsschlüssel für Ihren Arduino, um Daten an einen SSV-Cloud- bzw. IoT-Backend-Server zu schicken bzw. von diesem Server per Webseite zu lesen.

Ändern Sie die Zahlenwerte der beiden Elemente per Cut & Paste [Kopieren und Einfügen] auf die gleichen Werte, die Sie für Ihren Browser-Webzugriff eingestellt haben. Übertragen Sie dazu einfach die Zahlenwerte aus den Browsereinstellungen in den Arduino-Code.





```

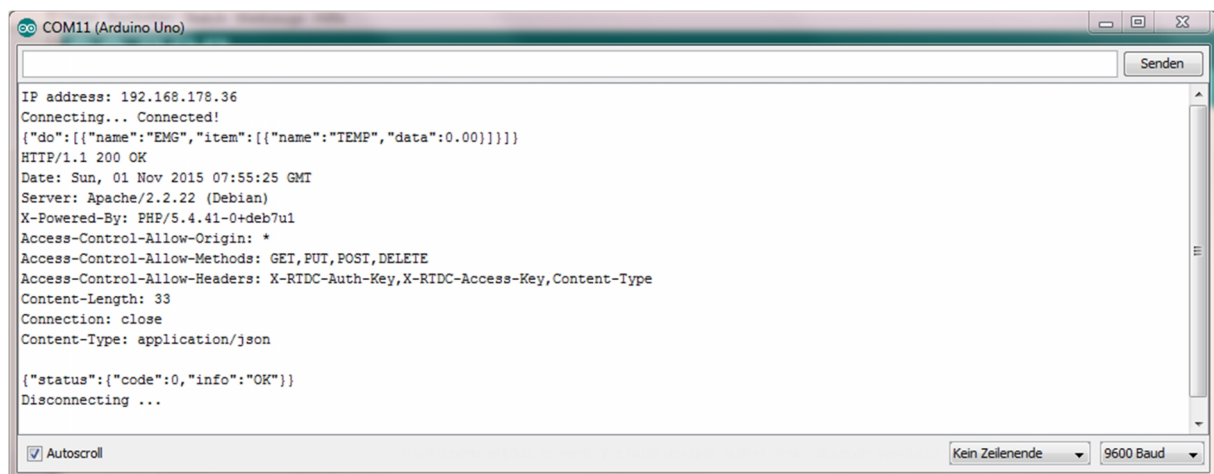
sketch_RTDC_REST_PUT_temp

rawOld = raw;

client.println("PUT /rtdc/v0/ HTTP/1.1");
client.println("Host: ssv.dynalias.net");
client.println("X-RTDC-Auth-Key: 8f810167-4711-4698-ae04-ce1e10998405");
client.println("X-RTDC-Access-Key: b393ceac-4411-4702-9228-8c60fc649812");
client.println("Content-Length: " + String(feedData.length()));
client.println("Connection: close");
client.println();
client.println(feedData);
}
else {
  Serial.println(" Connection failed!");
}
  
```

Übersetzen Sie den Arduino-Quellcode in der der IDE und laden Sie den ausführbaren Code in Ihren Arduino Uno.

Öffnen Sie in der Arduino-IDE den seriellen Monitor und beobachten Sie die Ausführung. Wenn alles OK ist, erhalten Sie in etwa die Ausgaben wie in der folgenden Abbildung.



```

COM11 (Arduino Uno)

IP address: 192.168.178.36
Connecting... Connected!
{"do":[{"name":"EMG","item":[{"name":"TEMP","data":0.00}]}]}
HTTP/1.1 200 OK
Date: Sun, 01 Nov 2015 07:55:25 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.41-0+deb7u1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,PUT,POST,DELETE
Access-Control-Allow-Headers: X-RTDC-Auth-Key,X-RTDC-Access-Key,Content-Type
Content-Length: 33
Connection: close
Content-Type: application/json

{"status":{"code":0,"info":"OK"}}
Disconnecting ...
  
```

Beobachten Sie die Temperaturanzeige im Browserfenster. Beachten Sie bitte, dass der Arduino nur bei einer Temperaturänderung einen neuen Wert an die Cloud übermittelt.

Viel Erfolg!

KDW / 0.3 / 03.11.2015