Erstellen Sie Eine Einfache ESP32-Wetterstation Mit BME280

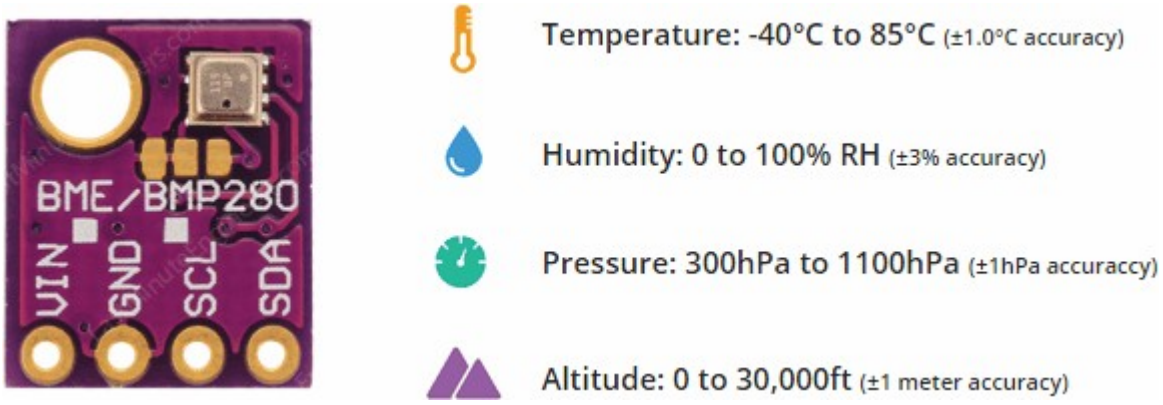
Lassen Sie sich nicht von der smartphone-Wetter-apps oder kommerzielle Wetterstationen(füttert Sie mit Daten von Stationen basierend Meilen entfernt) ruinieren Ihre outdoor-Pläne. Mit diesem IoT-Projekt Sie Ihre eigenen Wetterfrosch!

Dieses Projekt verwendet ESP32 als die control Gerät, dass leicht verbindet zu den bestehenden WiFi Netzwerk und erstellt ein Web-Server. Wenn jedes angeschlossene Gerät greift auf diese web-server, ESP32 liest in Temperatur, Luftfeuchtigkeit, Luftdruck und Höhe von BME280 & sendet es an den web-browser des Gerätes mit einem schönen interface. Aufgeregt? Let ' s get started!

BME280 Temperatur, Luftfeuchtigkeit und Druck-Sensor

Erste, werfen wir einen kurzen Blick auf die BME280-Modul.

BME280 ist die nächste generation von digitalen Temperatur -, Feuchte-und Druck-sensor, manufactured by Bosch. Es ist ein Nachfolger von sensoren wie BMP180, BMP085 oder BMP183.



Die Betriebsspannung des BME280-Modul ist von **3,3 V zu 5V** – Perfekt für eine Schnittstelle mit 3,3 V Mikrocontrollern wie ESP32.

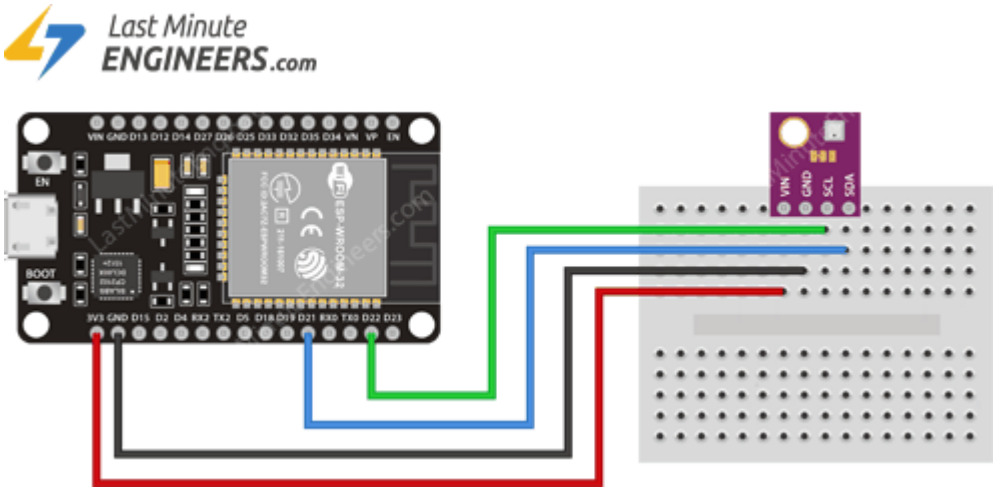
Das Modul verfügt über eine einfache zwei-Draht **I2C-Schnittstelle** für die Kommunikation. Die Standard I2C Adresse des BME280 Modul ist **0x76** und kann geändert werden, um 0x77 leicht mit diesem Verfahren .

Verdrahtung BME280 Sensor ESP32

Verbindungen sind ziemlich einfach. Starten durch Anschluss **der VIN** - pin auf die 3,3-V-Ausgang auf dem ESP32 und verbinden **GND** zu Boden.

Als Nächstes Verbinden Sie den **SCL** - pin mit dem I2C clock - **D22** - pin auf Ihrem ESP32 und verbinden Sie den **SDA** - pin mit dem I2C-Daten **D21** - pin auf Ihrem ESP32.

Das folgende Diagramm zeigt, wie Sie alles Draht.



Vorbereitung der Arduino IDE

Es gibt ein add-on für die Arduino IDE für die Programmierung des ESP32 mit der Arduino IDE. Folgen Sie den unten Tutorials bereiten Sie Ihre Arduino IDE zu arbeiten mit die ESP32, wenn Sie nicht bereits getan haben.

Einblick In ESP32 Funktionen & Verwendung Mit der Arduino IDE

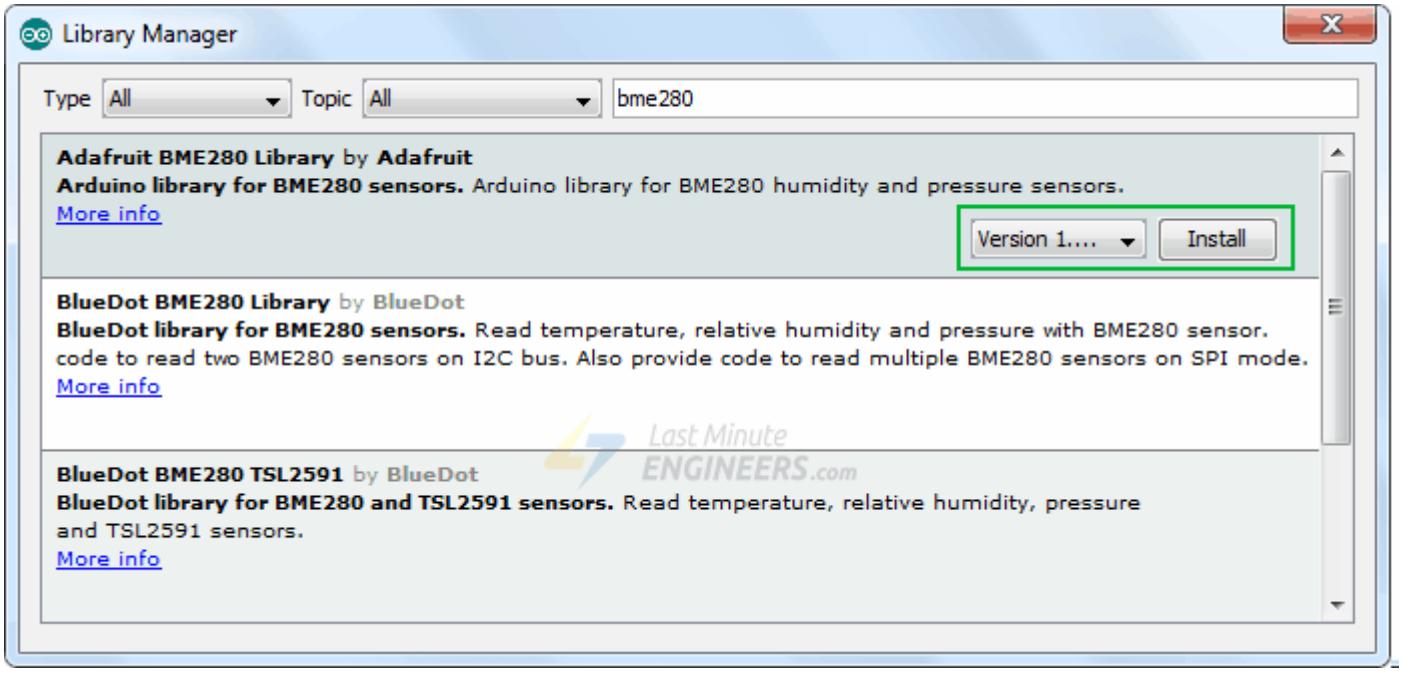
Paar Jahre zurück, ESP8266 nahm die embedded-IoT-Welt durch Sturm. Für weniger als $3, Sie erhalten könnte ein programmierbarer WLAN-fähigen mikrocontroller zu können...

Installation der Bibliothek Für BME280

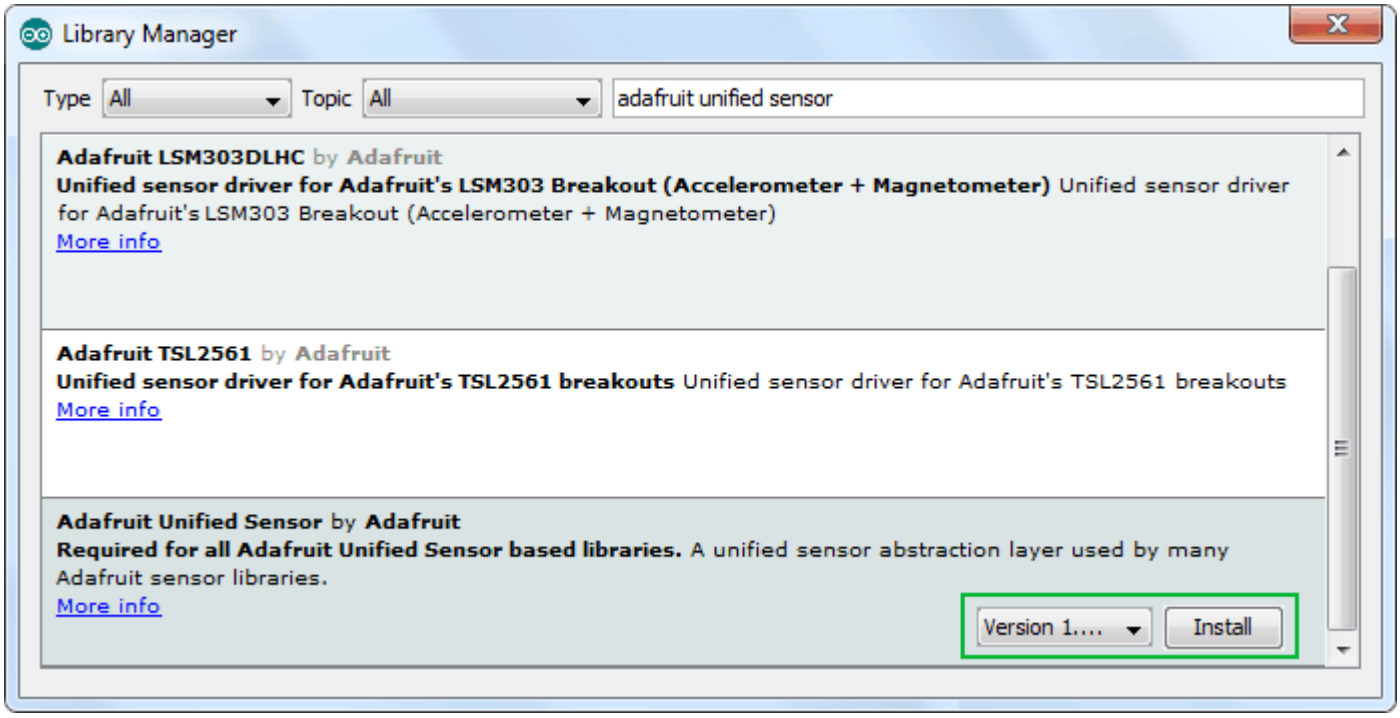
Die Kommunikation mit einem BME280 Modul ist ein Haufen Arbeit. Glücklicherweise Adafruit BME280 Library geschrieben wurde, sich zu verstecken alle die Komplexität, so dass wir können Problem die einfache Befehle zu Lesen Sie die Temperatur, relative Luftfeuchtigkeit & Luftdruck Daten.

Um die Bibliothek zu installieren navigieren Sie zu der **Arduino IDE - > Sketch > Include Library > Manage Libraries...** Warten-Bibliothek-Manager zum herunterladen von Bibliotheken-index und update Liste der installierten Bibliotheken.

Filtern Sie Ihre Suche durch Eingabe von ' **bme280** '. Es sollte ein paar Einträge. Suchen **Adafruit BME280 Bibliothek** von **Adafruit** . Klicken Sie auf diesen Eintrag, und wählen Sie dann Installieren aus.



Das BME280 sensor-Bibliothek verwendet die Adafruit-Sensor Unterstützung backend . Also, suchen Sie im Bibliothek-manager für den **Adafruit Unified Sensor** und installieren Sie das auch (Sie müssen möglicherweise scrollen bit)



Anzeige von Temperatur, Luftfeuchtigkeit, Druck und Höhe Auf ESP32 Web Server

Nun, wir konfigurieren unsere ESP32 in die Station (STA) Modus, und erstellen Sie ein web-server für das bereitstellen von web-Seiten zu jedem verbundenen client im Rahmen der bestehenden Netzwerk.

Wenn Sie wollen zu erfahren Sie mehr über die Erstellung einer web-server mit ESP32 in AP/STA Modus, check this tutorial out.

Erstellen Sie Eine Einfache ESP32 Web Server-Arduino IDE

Der neu eingeführte Nachfolger des ESP8266 - der ESP32 ist ein wachsender star unter den IoT or WiFi-related projects. Es ist eine sehr kostengünstige WiFi-Modul...

Bevor Sie den Kopf für das hochladen der Skizze, die Sie brauchen, um **eine Veränderung** zu machen es für Sie arbeiten. Sie ändern müssen die folgenden zwei Variablen mit Ihren Netzwerk-Anmeldeinformationen, so dass ESP32 kann eine Verbindung mit dem bestehenden Netzwerk.

```
const char* ssid = "YourNetworkName";  // Enter SSID here
const char* password = "YourPassword";  //Enter Password here
```

Sobald Sie fertig sind, gehen Sie voran und versuchen Sie die Skizze aus.

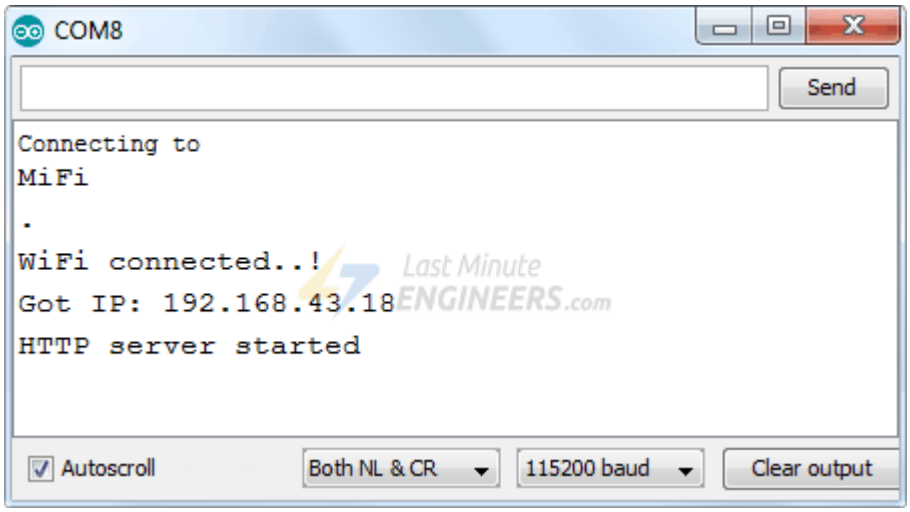
```
#include <WiFi.h>
```

```
#include <WebServer.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme;

float temperature, humidity, pressure, altitude;

/*Put your SSID & Password*/
const char* ssid = "YourNetworkName";  // Enter SSID here
const char* password = "YourPassword";  //Enter Password here

WebServer server(80);

void setup() {
  Serial.begin(115200);
  delay(100);

  bme.begin(0x76);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  //connect to your local wi-fi network
  WiFi.begin(ssid, password);

  //check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");

}
void loop() {
  server.handleClient();
}

void handle_OnConnect() {
  temperature = bme.readTemperature();
  humidity = bme.readHumidity();
  pressure = bme.readPressure() / 100.0F;
  altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);
  server.send(200, "text/html",
SendHTML(temperature,humidity,pressure,altitude));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

String SendHTML(float temperature,float humidity,float pressure,float altitude){
  String ptr = "<!DOCTYPE html> <html>\n";
```

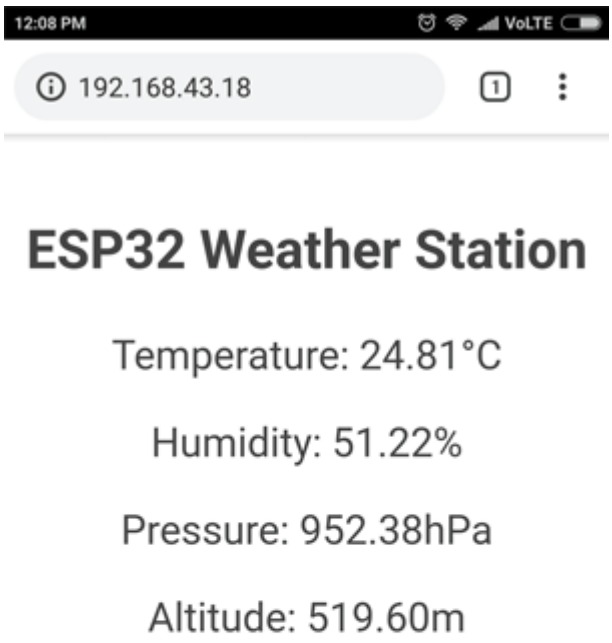
```
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, user-scalable=no\">\n";
  ptr +="<title>ESP32 Weather Station</title>\n";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin:
0px auto; text-align: center;}\n";
  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
  ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<div id=\"webpage\">\n";
  ptr +="<h1>ESP32 Weather Station</h1>\n";
  ptr +="<p>Temperature: ";
  ptr +=temperature;
  ptr +="&deg;C</p>";
  ptr +="<p>Humidity: ";
  ptr +=humidity;
  ptr +="%</p>";
  ptr +="<p>Pressure: ";
  ptr +=pressure;
  ptr +="hPa</p>";
  ptr +="<p>Altitude: ";
  ptr +=altitude;
  ptr +="m</p>";
  ptr +="</div>\n";
  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

Zugriff auf die Web-Server

Nach dem hochladen der Skizze, öffnen Sie den Seriellen Monitor mit einer Baudrate von 115200. Und drücken Sie die EN-Schaltfläche auf dem ESP32. Wenn alles OK ist, es wird Ausgang die dynamische IP-Adresse erhalten Sie von Ihrem router und zeigen, **HTTP-server gestartet** - Nachricht.



Anschließend laden Sie einen browser und zeigen Sie auf die IP-Adresse angezeigt, die auf den serial monitor. Der ESP32 sollte dazu dienen, eine web-Seite, die zeigen Temperatur, Luftfeuchtigkeit, Druck und Höhe von BME280.



Detaillierte Erläuterungen Zum Code

Die Skizze beginnt, indem Sie folgenden Bibliotheken.

- **WiFi.h** - Bibliothek bietet ESP32 bestimmten WiFi-Methoden, die wir aufrufen, um eine Verbindung zum Netzwerk.
- **WebServer.h** - Bibliothek verfügt über einige Methoden zur Verfügung, die uns helfen wird, einen server einzurichten und die Verarbeitung eingehender HTTP-Anforderungen, ohne zu sorgen über die low-level implementation details.
- **Draht.h** - Bibliothek kommuniziert mit jedem I2C-Gerät nicht nur die BME280.
- **Adafruit_BME280.h & Adafruit_Sensor.h** - Bibliotheken sind hardware-spezifische Bibliotheken die Griffe lower-level-Funktionen.

```
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

Als Nächstes erstellen wir ein Objekt vom sensor, und Variablen zum speichern von Temperatur, Luftfeuchtigkeit, Druck und Höhe.

```
#define SEALEVELPRESSURE_HPA (1013.25)
```

```
Adafruit_BME280 bme;

float temperature, humidity, pressure, altitude;
```

Als wir ESP32 Konfiguration in die Station (STA) - Modus, es wird ein Beitritt zu bestehenden WLAN-Netzwerk. Daher, wir müssen es mit Ihren Netzwerk-SSID & Password. Als Nächstes starten Sie web server auf port 80.

```
/*Put your SSID & Password*/
const char* ssid = "YourNetworkName";  // Enter SSID here
const char* password = "YourPassword";  //Enter Password here

WebServer server(80);
```

Innerhalb Von Setup () - Funktion

Innerhalb von Setup () - Funktion konfigurieren wir unseren HTTP-server, bevor Sie Sie tatsächlich ausführen.

Zunächst initialisieren wir die serielle Kommunikation mit PC und initialisieren BME-Objekt mit `begin()` Funktion. Es initialisiert I2C-Schnittstelle mit bestimmten I2C-Adresse(0x76) und prüft, ob die chip-ID korrekt ist. Dann setzt der chip mit soft-reset & wartet der sensor für die Kalibrierung nach dem aufwachen.

```
Serial.begin(115200);
delay(100);

bme.begin(0x76);
```

Nun, wir brauchen zu verbinden die WiFi-Netzwerk mit `WiFi.begin()` Funktion. Die Funktion nimmt SSID (Netzwerk-Name) und Kennwort als parameter.

```
Serial.println("Connecting to ");
Serial.println(ssid);

//connect to your local wi-fi network
WiFi.begin(ssid, password);
```

Während der ESP32 versucht, eine Verbindung zum Netzwerk herstellen, können wir überprüfen Sie den Verbindungsstatus mit WiFi.status() Funktion.

```
//check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
```

Sobald das ESP32 mit dem Netzwerk verbunden ist, wird die Skizze druckt die IP-Adresse zugewiesen zu ESP32 durch die Anzeige `WiFi.localIP()` Wert auf dem seriellen monitor.

```
Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
```

Um, um eingehende HTTP-Anforderungen, müssen wir angeben, welcher code ausgeführt werden soll, wenn eine URL ist der hit. Zu tun, verwenden wir **auf** Methode. Diese Methode nimmt zwei

Parameter. Erste ein URL-Pfad und das zweite ist der name der Funktion, die wir ausführen möchten, wenn die URL Treffer.

Der folgende code zeigt an, dass, wenn ein server eine HTTP-Anforderung empfängt, die auf das root - (/) Weg ist, wird es ausgelöst die `handle_OnConnect` Funktion. Beachten Sie, dass die angegebene URL ist ein relativer Pfad.

```
server.on("/", handle_OnConnect);
```

Wir haben noch nicht angegeben, was der server tun sollte, wenn der client fordert alle anderen URL als angegeben `server.on`. Es sollte Antworten mit einem HTTP-status 404 (Nicht Gefunden) und eine Nachricht für den Benutzer. Wir setzen dies in eine Funktion als gut, und verwenden Sie `server.onNotFound` zu sagen, dass es ausgeführt werden soll, wenn es erhält eine Anfrage für eine URL, war nicht angegeben `server.on`

```
server.onNotFound(handle_NotFound);
```

Nun, beginnen unsere server, nennen wir die begin-Methode auf dem server-Objekt.

```
server.begin();
Serial.println("HTTP server started");
```

Inside Loop () - Funktion

Die Verarbeitung der eigentlichen eingehende HTTP-Anforderungen, die wir benötigen, rufen Sie die `handleClient()` - Methode auf dem server-Objekt.

```
server.handleClient();
```

Neben, wir müssen eine Funktion erstellen, die wir befestigt, um root - (/) die URL mit `server.on` Erinnern Sie sich?

Beim start dieser Funktion ist, erhalten wir die Temperatur, Luftfeuchtigkeit, Druck und Höhe Werte von der sensor. In Auftrag zu reagieren, um die HTTP-Anforderung verwenden wir die **send** - Methode. Auch wenn die Methode aufgerufen werden kann, mit einem anderen Satz von Argumenten, seiner einfachsten form besteht der HTTP-Antwort-code, den Inhalts-Typ und den Inhalt.

In unserem Fall, wir werden senden Sie die code **200** (eines der HTTP-status-codes), das entspricht der **OK** - Antwort. Dann sind wir die Angabe der content-Typ als “text/html“, und schließlich fordern wir `SendHTML()` benutzerdefinierte Funktion, die erstellt eine dynamische HTML-Seite mit Temperatur -, Feuchte -, Druck-und Höhenangaben.

```
void handle_OnConnect() {
  temperature = bme.readTemperature();
  humidity = bme.readHumidity();
  pressure = bme.readPressure() / 100.0F;
  altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);
  server.send(200, "text/html",
SendHTML(temperature,humidity,pressure,altitude));
}
```

Ebenso müssen wir eine Funktion erstellen, um Griff 404 Fehler-Seite.

```
void handle_NotFound(){
```

```
  server.send(404, "text/plain", "Not found");
}
```

Die Anzeige des HTML-Web-Seite

`SendHTML()` Funktion ist verantwortlich für die Erzeugung einer web-Seite, wenn der ESP32 web server erhält eine Anfrage von einem web-client. Es werden lediglich verkettet HTML-code in eine große Zeichenfolge und gibt die `server.send()` Funktion, die wir früher diskutiert. Die Funktion Temperatur -, Feuchte -, Druck-und Höhenangaben als parameter zur dynamischen Generierung von HTML-Inhalten.

Der erste text, den Sie sollten immer senden die <!DOCTYPE> - Deklaration, die angibt, dass wir versenden HTML-code.

```
String SendHTML(float temperature,float humidity,float pressure,float altitude){
  String ptr = "<!DOCTYPE html> <html>\n";
```

Neben, die <meta> - viewport-element macht die web-Seite zu reagieren, in jedem web-browser, während der Titel-tag wird der Titel der Seite.

```
ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, user-scalable=no\">\n";
ptr +="<title>ESP32 Weather Station</title>\n";
```

Styling der Web-Seite

Neben, wir haben einige CSS-Stil der web-Seite Aussehen. Wählen wir die Helvetica-Schrift, definieren die Inhalte angezeigt werden als inline-block ausgerichtet und in der Mitte.

```
ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}\n";
```

Folgende code legt die Farbe, schriftart und-Marge um den Körper, H1 und p-tags.

```
ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
ptr +="</style>\n";
ptr +="</head>\n";
ptr +="<body>\n";
```

Die Einstellung der Web-Seite Überschrift

Als Nächstes eine überschrift für die web-Seite eingestellt ist; Sie können diesen text ändern, um alles, was Anzüge Ihre Anwendung.

```
ptr +="<div id=\"webpage\">\n";
ptr +="<h1>ESP32 Weather Station</h1>\n";
```

Anzeige der Messwerte auf Web-Seite

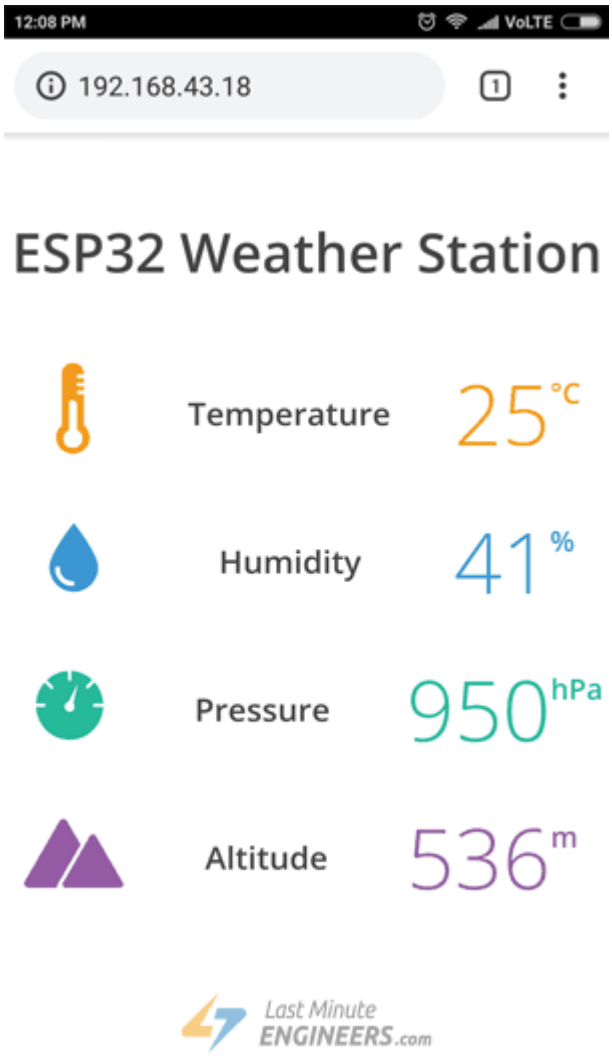
Die dynamische Darstellung der Temperatur -, Feuchte -, Druck-und Höhenangaben, fügen wir diese Werte in Absatz-Tags. Zur Anzeige von Grad-symbol verwenden wir HTML-Entität **°**

```
ptr +="<p>Temperature: ";
ptr +=temperature;
ptr +="&deg;C</p>";
```

```
ptr +="<p>Humidity: ";
ptr +=humidity;
ptr +="%</p>";
ptr +="<p>Pressure: ";
ptr +=pressure;
ptr +="hPa</p>";
ptr +="<p>Altitude: ";
ptr +=altitude;
ptr +="m</p>";
ptr +="</div>\n";
ptr +="</body>\n";
ptr +="</html>\n";
return ptr;
}
```

Styling Web-Seite zu Professioneller Aussehen

Programmierer wie uns oft eingeschüchtert durch design – aber ein wenig Aufwand können Sie Ihre web-Seite Aussehen attraktiver und professioneller. Unten screenshot wird Ihnen eine grundlegende Idee von dem, was wir tun werden.



Ziemlich erstaunlich, Richtig? Ohne weitere Umschweife, lassen Sie uns einige Stil zu unserem vorherigen HTML-Seite. Um zu beginnen mit, kopieren und fügen Sie folgenden code zu ersetzen `SendHTML()` Funktion von der Skizze oben.

```
String SendHTML(float temperature,float humidity,float pressure,float altitude){
  String ptr = "<!DOCTYPE html>";
  ptr +="<html>";
  ptr +="<head>";
  ptr +="<title>ESP32 Weather Station</title>";
  ptr +="<meta name='viewport' content='width=device-width, initial-
scale=1.0'>";
  ptr +="<link href='https://fonts.googleapis.com/css?
family=Open+Sans:300,400,600' rel='stylesheet'>";
  ptr +="<style>";
  ptr +="html { font-family: 'Open Sans', sans-serif; display: block; margin:
0px auto; text-align: center;color: #444444;}";
  ptr +="body{margin: 0px;} ";
  ptr +="h1 {margin: 50px auto 30px;} ";
  ptr +=".side-by-side{display: table-cell;vertical-align: middle;position:
relative;}";
  ptr +=".text{font-weight: 600;font-size: 19px;width: 200px;}";
  ptr +=".reading{font-weight: 300;font-size: 50px;padding-right: 25px;}";
  ptr +=".temperature .reading{color: #F29C1F;}";
  ptr +=".humidity .reading{color: #3B97D3;}";
  ptr +=".pressure .reading{color: #26B99A;}";
  ptr +=".altitude .reading{color: #955BA5;}";
  ptr +=".superscript{font-size: 17px;font-weight: 600;position: absolute;top:
10px;}";
  ptr +=".data{padding: 10px;}";
  ptr +=".container{display: table;margin: 0 auto;}";
  ptr +=".icon{width:65px}";
  ptr +="</style>";
  ptr +="</head>";
  ptr +="<body>";
  ptr +="<h1>ESP32 Weather Station</h1>";
  ptr +="<div class='container'>";
  ptr +="<div class='data temperature'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 19.438 54.003'height=54.003px
id=Layer_1 version=1.1 viewBox='0 0 19.438 54.003'width=19.438px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063
v30.982";
  ptr
+="C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.719
-4.351,9.719-9.718";
  ptr +="c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-
2h4.084V8.82H11.976z M15.302,44.833";
  ptr +="c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-
4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22";
  ptr
+="s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fill
=#F29C21 /></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Temperature</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)temperature;
  ptr +="<span class='superscript'>&deg;C</span></div>";
  ptr +="</div>";
  ptr +="<div class='data humidity'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 29.235 40.64'height=40.64px id=Layer_1
version=1.1 viewBox='0 0 29.235 40.64'width=29.235px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink
y=0px><path
d='M14.618,0C14.618,0,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.617-
6.544,14.617-14.617";
```

```
  ptr +="C29.235,17.95,14.618,0,14.618,0z M13.667,37.135c-5.604,0-10.162-4.56-
10.162-10.162c0-0.787,0.638-1.426,1.426-1.426";
  ptr
+="c0.787,0,1.425,0.639,1.425,1.426c0,4.031,3.28,7.312,7.311,7.312c0.787,0,1.425
,0.638,1.425,1.425";
  ptr +="C15.093,36.497,14.455,37.135,13.667,37.135z'fill=#3C97D3 /></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Humidity</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)humidity;
  ptr +="<span class='superscript'>%</span></div>";
  ptr +="</div>";
  ptr +="<div class='data pressure'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 40.542 40.541'height=40.541px
id=Layer_1 version=1.1 viewBox='0 0 40.542 40.541'width=40.542px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M34.313,20.271c0-
0.552,0.447-1,1-1h5.178c-0.236-4.841-2.163-9.228-5.214-12.593l-3.425,3.424";
  ptr +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-0.293c-0.391-
0.391-0.391-1.023,0-1.414l3.425-3.424";
  ptr +="c-3.375-3.059-7.776-4.987-12.634-
5.215c0.015,0.067,0.041,0.13,0.041,0.202v4.687c0,0.552-0.447,1-1,1s-1-0.448-1-
1V0.25";
  ptr +="c0-0.071,0.026-0.134,0.041-
0.202C14.39,0.279,9.936,2.256,6.544,5.385l3.576,3.577c0.391,0.391,0.391,1.024,0,
1.414";
  ptr +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-
0.293L5.142,6.812c-2.98,3.348-4.858,7.682-5.092,12.459h4.804";
  ptr +="c0.552,0,1,0.448,1,1s-0.448,1-
1,1H0.05c0.525,10.728,9.362,19.271,20.22,19.271c10.857,0,19.696-8.543,20.22-
19.271h-5.178";
  ptr +="C34.76,21.271,34.313,20.823,34.313,20.271z M23.084,22.037c-0.559,1.561-
2.274,2.372-3.833,1.814";
  ptr +="c-1.561-0.557-2.373-2.272-1.815-3.833c0.372-1.041,1.263-1.737,2.277-
1.928L25.2,7.202L22.497,19.05";
  ptr +="C23.196,19.843,23.464,20.973,23.084,22.037z'fill=#26B999 /></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Pressure</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)pressure;
  ptr +="<span class='superscript'>hPa</span></div>";
  ptr +="</div>";
  ptr +="<div class='data altitude'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 58.422 40.639'height=40.639px
id=Layer_1 version=1.1 viewBox='0 0 58.422 40.639'width=58.422px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M58.203,37.754l0.007-
0.004L42.09,9.935l-0.001,0.001c-0.356-0.543-0.969-0.902-1.667-0.902";
  ptr +="c-0.655,0-1.231,0.32-1.595,0.808l-0.011-0.007l-0.039,0.067c-0.021,0.03-
0.035,0.063-0.054,0.094L22.78,37.692l0.008,0.004";
  ptr +="c-0.149,0.28-0.242,0.594-
0.242,0.934c0,1.102,0.894,1.995,1.994,1.995v0.015h31.888c1.101,0,1.994-
0.893,1.994-1.994";
  ptr +="C58.422,38.323,58.339,38.024,58.203,37.754z'fill=#955BA5 /><path
d='M19.704,38.674l-0.013-0.004l13.544-23.522L25.13,1.156l-
0.002,0.001C24.671,0.459,23.885,0,22.985,0";
  ptr +="c-0.84,0-1.582,0.41-2.051,1.038l-0.016-0.01L20.87,1.114c-0.025,0.039-
0.046,0.082-0.068,0.124L0.299,36.851l0.013,0.004";
  ptr
+="C0.117,37.215,0,37.62,0,38.059c0,1.412,1.147,2.565,2.565,2.565v0.015h16.989c-
0.091-0.256-0.149-0.526-0.149-0.813";
  ptr +="C19.405,39.407,19.518,39.019,19.704,38.674z'fill=#955BA5 /></g></svg>";
```

```
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Altitude</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)altitude;
  ptr +="<span class='superscript'>m</span></div>";
  ptr +="</div>";
  ptr +="</div>";
  ptr +="</body>";
  ptr +="</html>";
  return ptr;
}
```

Wenn Sie versuchen zu vergleichen Sie diese Funktion mit dem vorhergehenden, Sie kommen zu wissen, dass Sie sind ähnlich, außer diese änderungen.

- Wir haben Google beauftragt, Open-Sans - schriftart, die für unsere web-Seite. Beachten Sie, dass Sie nicht sehen können, Google font, ohne aktive internet-Verbindung auf das Gerät. Google fonts geladen werden auf die Fliegen.

  ```
  ptr +="<link href='https://fonts.googleapis.com/css?
  family=Open+Sans:300,400,600' rel='stylesheet'>";
  ```

- Die verwendeten Symbole zur Anzeige von Temperatur, Luftfeuchtigkeit, Druck und Höhe Lesungen sind eigentlich eine Scalable Vector Graphics (SVG) definiert in <svg> - tag. Erstellen von SVG-man braucht keine speziellen Programmierkenntnisse. Sie verwenden können Google-SVG-Editor für die Erstellung von Grafiken für Ihre Seite. Wir haben diese SVG-Symbole.

Verbesserung der Code – Auto Page Refresh

Eine der Verbesserungen, die Sie tun können mit unserem code ist die Seite aktualisieren automatisch, um die änderung der sensor Wert.

Mit dem Zusatz von einer einzigen meta-tag in Ihre HTML-Dokument, können Sie der browser automatisch neu laden der Seite zu einer vorgesehenen Intervall.

```
<meta http-equiv="refresh" content="2" >
```

Platzieren Sie diesen code in den <head> - tag des Dokuments, das meta-tag weist den browser zu aktualisieren, der alle zwei Sekunden. Ziemlich clever!

Das dynamische laden von Sensor-Daten mit AJAX

Aktualisieren einer web-Seite ist nicht zu praktisch, wenn Sie einen schweren web-Seite. Eine bessere Methode ist die Verwendung von Asynchronous Javascript And Xml (**AJAX**), so dass

können wir Daten anfordern, die vom server asynchron (im hintergrund), ohne die Seite aktualisieren.

Das XMLHttpRequest - Objekt in JavaScript wird Häufig zum ausführen von AJAX auf Webseiten. Es führt die silent-GET-Anforderung auf dem server und aktualisiert das element auf der Seite. AJAX ist keine neue Technologie, oder andere Sprache, nur bestehende Technologien auf neue Weise. Neben dieser, AJAX macht es auch möglich,

- Anfordern von Daten von einem server aus, nachdem die Seite geladen wurde
- Empfangen Sie Daten von einem server aus, nachdem die Seite geladen wurde
- Senden von Daten an einen server in den hintergrund

Hier ist der AJAX-Skript, das wir verwenden werden. Platzieren Sie dieses Skript nur, bevor Sie Sie schließen </head> - tag.

```
ptr +="<script>\n";
ptr +="setInterval(loadDoc,1000);\n";
ptr +="function loadDoc() {\n";
ptr +="var xhttp = new XMLHttpRequest();\n";
ptr +="xhttp.onreadystatechange = function() {\n";
ptr +="if (this.readyState == 4 && this.status == 200) {\n";
ptr +="document.body.innerHTML =this.responseText}\n";
ptr +="};\n";
ptr +="xhttp.open(\"GET\", \"/\", true);\n";
ptr +="xhttp.send();\n";
ptr +="}\n";
ptr +="</script>\n";
```

Das Skript beginnt mit <script> - tag. Als AJAX-Skript ist nichts anderes als eine javascript -, müssen wir schreiben es in <script> - tag. Um diese Funktion wiederholt aufgerufen, die wir verwenden werden, der javascript `setInterval()` Funktion. Es nimmt zwei Parameter eine Funktion, die ausgeführt werden, und Zeit-Intervall (in Millisekunden) an, wie oft die Funktion ausführen.

```
ptr +="<script>\n";
ptr +="setInterval(loadDoc,1000);\n";
```

Das Herz dieses Skript ist eine `loadDoc()` Funktion. Innerhalb dieser Funktion, ein `XMLHttpRequest()` Objekt erstellt wird. Dieses Objekt dient zum anfordern von Daten von einem web-server.

```
ptr +="function loadDoc() {\n";
ptr +="var xhttp = new XMLHttpRequest();\n";
```

Die `xhttp.onreadystatechange()` Funktion wird jedesmal aufgerufen, wenn der readyState ändert. Die readyState-Eigenschaft enthält den status des XMLHttpRequest. Es hat einen der folgenden Werte.

- 0: Anforderung nicht initialisiert
- 1: Verbindung zum server hergestellt
- 2: Anforderung erhalten
- 3: Bearbeitung der Anfrage
- 4: Anforderung abgeschlossen und die Antwort ist bereit

Die status-Eigenschaft enthält den status des XMLHttpRequest-Objekt. Es hat einen der folgenden Werte.

- 200: "OK"
- 403: Verboten"""
- 404: "Seite nicht gefunden"

Wenn der readyState 4 und der status ist 200, die Antwort ist fertig. Nun, der Inhalt von **Körper** (holding Temperatur-Messwerte) aktualisiert wird.

```
ptr +="xhttp.onreadystatechange = function() {\n";
ptr +="if (this.readyState == 4 && this.status == 200) {\n";
ptr +="document.body.innerHTML =this.responseText}\n";
ptr +="};\n";
```

Der HTTP-request wird dann eingeleitet, die über die open() und send () - Funktionen.

```
ptr +="xhttp.open(\"GET\", \"/\", true);\n";
ptr +="xhttp.send();\n";
ptr +="}\n";
```