# Hard- en Software Ontwerp van Webinterface dat Temperatuur en Luchtvochtigheid *Realtime* Presenteert

Semih C Karakoç
*Computer Science and Engineering*
Inholland University of Applied Sciences
Alkmaar, Netherlands
Email: 695258@student.inholland.nl

***Samenvatting*—In dit artikel is onderzocht, hoe *realtime* temperatuur en luchtvochtigheid meetgegevens op een juiste wijze op een webinterface gepresenteerd kunnen worden. Daarnaast hoe de temperatuur en luchtvochtigheidsensor werkt, welke microcontroller geschikt is voor deze toepassing. Tenslotte wordt de hoofdvraag beantwoord.**

***Index Terms*—MCU, DHT11, hardware, software, C++, Arduino, SoftAP, Wi-Fi, webinterface**

## I. Inleiding

Het vastleggen van omgevingseigenschappen, zoals luchtvochtigheid en temperatuur speelt een rol in de verschillende domeinen en worden bijvoorbeeld toegepast in consumptiegoederen, testapparatuur, verwarming, ventilatie en airconditioning [1]. Een veelgebruikt digitaal component voor het meten van luchtvochtigheid en temperatuur is de "DHT-sensor" (*Digital Humidity Temperature* sensor) [2]. Er bestaan verschillende soorten DHT-sensoren, maar in dit artikel wordt voornamelijk verwezen naar de DHT versie als het gaat om DHT11-sensor.

In dit artikel wordt specifiek ingegaan op het ontwerpen van hard- en software met een DHT-sensor in combinatie met een MCU unit (MCU), die de gemeten temperatuur en luchtvochtigheid waarden *realtime* kan presenteren op een *end device* met behulp van een webinterface.

Het probleem dat hier speelt betreft het continu meten van de *realtime* meetwaarden van een ruimte en deze op een toegankelijke wijze te presenteren via een webinterface.

Het technische probleem kan dan vertaald worden als het overdragen van de meetgegevens aan een MCU die deze *realtime* meetgegevens via een op te zetten webinterface meetgegevens kan visualiseren middels een *end device*.

De relevantie van het oplossen van dit technische probleem is dat het gebruikers in staat stelt om op afstand (*wireless*) en *realtime* de luchtvochtigheid en temperatuur gegevens van een ruimte afgelezen kunnen worden. Dit kan bijvoorbeeld toegepast worden met een IoT (*Internet of Things*) toepassing in de landbouw industrie, waar nauwkeurige realtimegegevens nodig kunnen zijn voor het nemen van beslissingen, bijvoorbeeld het optimaliseren van landbouwprestaties [3].

Het doel van dit onderzoek betreft oplossen van technische probleem met een werkend systeem. Om dit doel te behalen is onderzoek gedaan toepassingsmogelijkheden van een MCU in combinatie met de DHT-sensor.

In het onderzoek is literatuur- en component specificatie documentatieonderzoek als methode gekozen.

Het onderzoeken van de werking en toepassingsmogelijkheden van de DHT-sensor, MCU en het implementeren van het visualiseren via een webinterface vormt het kern van dit onderzoek. De technische aspecten en werking ervan worden in dit artikel besproken.

### A. Overzicht

De lezer zal door dit artikel in de volgende volgorde worden begeleid:

- ➢ paragraaf II: *Probleemstelling* worden de hoofdvraag en deelvragen geformuleerd;
- ➢ paragraaf III: *Methoden* worden de toegepaste methoden uitgebreid uitgelegd;
- ➢ paragraaf IV: *Resultaten* worden de resultaten van de gebruikte methode uitgewerkt;
- ➢ paragraaf V: *Conclusies* worden conclusies getrokken uit de resultaten;
- ➢ paragraaf VI: *Discussie* wordt een reflectie gedaan op de opgezette onderzoek.
- ➢ paragraaf VII: *Aanbevelingen* worden aanbevelingen gedaan voor een mogelijke vervolgonderzoek.

## II. Probleemstelling

Een DHT-sensor gecombineerd met een MCU zou het monitoren van luchtvochtigheid en temperatuur in diverse toepassingsgebieden, als een simpele oplossing mogelijk kunnen maken, zoals in een huiskamer. Helaas ontbreekt een simpele

hard- en software ontwerp om de gegevens vanuit DHT-sensor te verzamelen en op een bepaalde manier deze data *realtime* aan een eindgebruiker beschikbaar te stellen middels een webinterface.

### A. Probleemanalyse

Het probleem kan worden beschreven als het ontbreken van een oplossing voor het realtime visualiseren van de meetgegevens van DHT-sensor via een webinterface. Om de spelende probleem te kanaliseren, is het nodig om de technische aspecten van de te gebruiken hardware te doorgronden en te analyseren.

Een DHT-sensor is een gecombineerde luchtvochtigheid en temperatuursensor die alleen hardware-matig meet en deze gegevens beschikbaar stelt. Technisch gezien ontbreekt aan deze sensor dataverwerking en de communicatie mogelijkheden om de meetgegevens te kunnen presenteren via een webinterface. Om dat mogelijk te maken is er een MCU als hardware nodig die de meetgegevens van de DHT-sensor uit kan lezen, verwerken en vervolgens presenteren via een webinterface. Een ander probleem hierbij betreft de benodigde software-ontwerp waarmee via een Wi-Fi netwerk en webinterface de meetgegevens gepresenteerd kunnen worden op een webpagina via een *end-device*. Echter, om dit probleem op een simpele en doeltreffende manier te tackelen is het een noodzaak om de programmeermogelijkheden en de connectiviteitsopties van de hardware (DHT en MCU) te kennen.

### B. Vooronderzoek

Het doel van dit vooronderzoek is het verzamelen van technische informatie over de DHT11-sensor. Daarnaast zal met dit vooronderzoek een definitie keuze gemaakt worden welke MCU toegepast kan worden om het probleem op te lossen. Daarvoor zijn meerdere literatuur-vooronderzoeken gedaan:

- Uit het eerste literatuuronderzoek [4] is gebleken dat de DHT11-sensor in staat is om temperatuur en luchtvochtigheid te meten, en voorzien is van een *calibrated digital signal* voor de uitgang-pen. De DHT11-sensor maakt gebruik van de *calibration coefficient* dat in het "OTP" (*One-Time-Programmable*) programmageheugen bewaard wordt. Wanneer de sensor de temperatuur- en vochtigheidsveranderingen meet, leest de sensor de *sensor coefficient* uit en voert deze een berekening uit. De DHT11-sensor wordt in klein formaat gefabriceerd, waardoor het praktisch in gebruik is. Ook hebben de onderzoekers van dat dezelfde artikel gebruik gemaakt van een ESP8266 MCU. De reden waarom ze deze MCU hebben gekozen is, omdat de ESP8266 een Wi-Fi chip bevat. De Wi-Fi chip is essentieel voor het opzetten van een Wi-Fi netwerk, dat gebruikers verbindingsmogelijkheden biedt om data af te lezen via een webserver. Daarnaast biedt deze MCU genoeg I/O-pennen, zodat die ingezet kunnen worden voor een *monitoring and controlling application*. Om deze MCU te programmeren vertellen de auteurs dat ze gebruik hebben gemaakt van het software ontwikkel taal en programma genaamd "Arduino".

  De auteurs hadden in hun artikel bijna dezelfde probleemstelling gedefinieerd als in dit artikel. Het betrof de volgende probleemstelling: "*Describing the problem clearly will help in designing and making a temperature and humidity monitoring system tool*". De auteurs hebben een aantal methoden toegepast, zoals literatuuronderzoek, deskresearch en hardware- en softwareontwerp.

- In de tweede literatuuronderzoek [5] wordt vermeld dat deze DHT11-sensor een complexe luchtvochtigheid en temperatuur meting bevat met een gekalibreerde digitale signaaluitvoer. Het betreft een gecombineerde module voor het waarnemen van vochtigheid en temperatuur. De DHT11-sensor bestaat uit een *resistive* luchtvochtigheids meetcomponent en een NTC *(Negative Temperature Coefficient)* temperatuur meetcomponent. De DHT11-sensor werkt op seriële communicatie, dat wil zeggen dat data verzonden wordt via één-draad communicatie. Het voordeel van één-draad communicatie is dat het minder vermogen verbruikt in vergelijking met parallel-draad communicatie. Het nadeel is dat één-draad communicatie langer over doet om data van punt $a$ naar punt $b$ te verzenden [6]. Het procestijd van deze sensor is ongeveer 4 milliseconden, daarnaast moet ook een initialisatie vertraging van één seconde ingesteld worden. De DHT11-sensor is klein van formaat, met een lage energieverbruik waarbij de signaaloverdracht circa tot 20 meter plaats kan vinden. Deze DHT11-sensor bevat vier I/O-pennen, namelijk: de $V_{cc}$-pen, de data-pen, de NC-pen (*Not Connected*) en de $GND$-pen.

- De derde literatuuronderzoek [7] gaat een stap verder in op de werking van de DHT11-sensor (kopje 3C van dat artikel). Onder dat hoofdstuk wordt uitgelegd dat de DHT11-sensor een temperatuur meetbereik heeft van 0 tot 50°C met een nauwkeurigheidsmarge van $\pm 2$°C. Het meetbereik van de luchtvochtigheidssensor is 20 tot 90% RH (*Relative Humidity*) met een nauwkeurigheidsmarge van $\pm 5$% RH. Ook wordt uitgelegd dat de DHT11-sensor een reageertijd heeft van 1 seconden, frequentie van 1Hz.

- In de laatste artikel [8] is er onderzocht hoe de temperatuur en luchtvochtigheid gepresenteerd kan worden op een web-server. Dit onderzoeksartikel gaat in op de technische specificaties van de DHT22-sensor en ESP32 MCU. Er wordt uitgelegd dat de ESP32 een *low-cost* en *low-power* MCU is. Deze MCU is gebruikt, omdat het een Wi-Fi chip bevat die gebruikt kan worden om een webserver op te zetten. Om de webserver op te zetten wordt er gebruik gemaakt van HTML/CSS/JS en ESP32 Arduino bibliotheken, en deze

software wordt ontwikkeld in de Arduino IDE omgeving.

➢ Deze vier artikelen geven een aantal oplossingsrichtingen die mogelijk van toepassing kunnen zijn voor dit onderzoek, namelijk:

- Het gebruik van een ESP32 MCU om een Wi-Fi netwerk op te zetten. Deze ESP32 MCU is gekozen, omdat dit een van de nieuwste Wi-Fi MCU is [9];
- Het bestuderen van de technische data specificaties van de DHT11-sensor, zodat op een juiste wijze een hardware ontwerp ontwikkeld kan worden;
- Het bestuderen en gebruiken van Arduino IDE en de bijbehorende bibliotheken, waarmee de standaard programmeermogelijkheden toegepast kunnen worden.

### C. Vraagstelling

De hoofdvraag is:

"*Hoe kan een werkend systeem ontwikkeld worden met een MCU die de realtime gemeten luchtvochtigheid en temperatuurgegevens door de DHT-sensor kan uitlezen, verwerken en met een eigen Wi-Fi netwerk deze gegevens op een toegankelijke wijze kan presenteren via een webinterface op diverse end-devices?*"

Om deze hoofdvraag te kunnen beantwoorden, zijn er drie deelvragen bedacht, namelijk:

*A*. Op welke simpele manier kunnen de meetgegevens van de DHT11-sensor realtime uitgelezen worden door een ESP32 MCU?

*B*. Verken op welke elementaire wijze een Wi-Fi netwerk opgezet kan worden met de ESP32 MCU?

*C*. Op welke wijze kunnen de meetgegevens continu ververst en gepresenteerd worden via een webinterface?

Deze deelvragen zijn gericht op het verkrijgen van inzicht in de technische mogelijkheden van de te gebruiken hardware (DHT11-sensor en ESP32 MCU) en de mogelijke oplossingsrichtingen voor het ontwikkelen van een simpel en werkend systeem. Het beantwoorden van deze deelvragen zal uiteindelijk moeten leiden tot een oplossing voor het visualiseren van *realtime* luchtvochtigheids- en temperatuurmetingen op een webpagina via een *end-device*.

## III. Methoden

Voor dit artikel is gekozen om als primaire methode een formeel ontwerp te realiseren.

### A. Op welke simpele manier kunnen de meetgegevens van de DHT11-sensor realtime uitgelezen worden door een ESP32 MCU?

Om deze deelvraag te beantwoorden wordt een literatuur- en technische-specificatie onderzoek gedaan om. Deze methode is een vorm van deskresearch.

Er is hiervoor gekozen, omdat de gebruikte hardware gebaseerd is op bepaalde communicatiemogelijkheden die de fabrikant voorgeschreven heeft. Literatuuronderzoek is belangrijk om informatie te verzamelen over het onderwerp, ook is kennis van andere eerdere gedane onderzoeken handig om paraat te hebben, zodat de deze deelvraag op een juiste manier beantwoord kan worden. De te doorlopen stappen hierbij zijn:

➢ Bestuderen van technische specificaties hoe de benodigde hardware op de juiste manier toegepast kunnen worden:

1) DHT11-sensor;
2) ESP32 MCU.

➢ Literatuuronderzoek naar het uitlezen van meetgegevens door een MCU.

Voor het hardware ontwerp zijn de technische specificaties essentieel om een werkend ontwerp te kunnen realiseren. Het is belangrijk om andere vergelijkbare ontwerpen die gepubliceerd zijn in andere artikelen door te lezen en om daarvan te leren en daarop verder door te bouwen.

Als resultaat is het van belang om een werkend systeem op te leveren waarmee realtime de gemeten waarden van de DHT11-sensor uitgelezen en verwerkt kunnen worden door de ESP32 MCU.

### B. Verken op welke elementaire wijze een Wi-Fi netwerk opgezet kan worden met de ESP32 MCU?

Om deze tweede deelvraag te beantwoorden wordt de ESP-IDF documentatie onderzocht (*deskresearch*).
Er is hiervoor gekozen, uit het vooronderzoek gebleken is dat de ESP32 MCU een aantal standaard mogelijkheden kent voor het opzetten van een Wi-Fi netwerk. De te doorlopen stappen hierbij zijn:

➢ Bestuderen van ESP-IDF documentatie ter verkennen van Wi-Fi mogelijkheden;

➢ Vergelijk de verschillende WiFi methoden en configuratie mogelijkheden van ESP32 MCU om daarna een juiste keuze te kunnen maken voor een elementaire Wi-Fi netwerk opzet;

➢ Realiseer met de gekozen Wi-Fi netwerk mogelijkheid de benodigde elementaire Wi-Fi netwerk.

Er is hiervoor gekozen om op een doeltreffende en simpele wijze gebruik te kunnen maken van de standaard mogelijkheden uit de beschikbare Wi-Fi oplossingen van de ESP32.

Er is voldoende resultaat geboekt als er op een simpele manier een werkende elementaire Wi-Fi netwerk gerealiseerd is met de ESP32 MCU.

*C. Op welke wijze kunnen de meetgegevens continu ververst en gepresenteerd worden via een webinterface?*

Om tenslotte de derde deelvraag te kunnen beantwoorden is een literatuuronderzoek gedaan.
Er is voor deze methode gekozen, omdat er ten eerste meerdere programmeermogelijkheden zijn en ten tweede is dit al eerder toegepast door andere onderzoekers. Het is essentieel om de webinterface ververst te houden, zodat de gemeten gegevens ook *realtime* gepresenteerd kunnen worden. De genomen stappen om deze deelvraag te kunnen beantwoorden zijn:

➢ Literatuuronderzoek naar mogelijke programmeermogelijkheden die gebruikt kunnen worden om een webpagina te verversen;

➢ Op welke wijze kan de gekozen programmeermogelijkheid geïmplementeerd worden en welke functies of methoden horen daar bij;

➢ Onderzoek hoe en welke taal geschikt is om de realtime data te presenteren via de webinterface.

De manier van aangepakhelpt enerzijds bij het gebruik kunnen maken van de bevindingen van de andere onderzoekers en anderzijds kan dit leiden tot een simpele en efficiënte manier een webinterface te verversen.

Het resultaat is succesvol te noemen als de uitgevoerde stappen leiden tot een efficiënte en effectieve oplossing voor deze deelvraag.

## IV. Resultaten

In dit hoofdstuk staan de resultaten van het onderzoek per deelvraag uitgewerkt. Er is voornamelijk literatuur- en component specifieke documentatieonderzoek als methode gebruikt.

*A. Op welke simpele manier kunnen de meetgegevens van de DHT11-sensor realtime uitgelezen worden door een ESP32 MCU?*

➢ Onderzoek van de technische specificaties van de DHT11-sensor

Volgens de datasheet [10] van de DHT11-sensor zijn de onderstaande gegevens belangrijk als uitgangspunt:

| **Specificaties** | **DHT11-11 sensor** |
|---|---|
| *Communicatie Protocol* | *Single-Wire Two-Way* |
| *Temperatuur meetbereik* | 0 - 50 °C |
| *Luchtvochtigheid meetbereik* | 20 - 90% RH |
| *Temperatuur nauwkeurigheid* | ±2°C |
| *Luchtvochtigheid nauwkeurigheid* | ±5% RH |
| *Voedingsbereik* | 3 - 5.5V DC |
| *"Sample period"* | 2 secondes |

Tabel I
Technische specificaties van de DHT11-sensor.

Deze sensor bevat een *resistive-type* luchtvochtigheidsmetingscomponent en een NTC-component voor de temperatuurmeting. De sensor heeft in totaal vier pennen, namelijk:

- Pen-1: De $V_{cc}$ ingang met een spanningsingang van 3.3V DC;
- Pen-2: De data pen. Deze kan worden aangesloten aan een van de GPIO (*General Purpose Input Output*) pennen van een MCU;
- Pen-3: Deze pen wordt niet gebruikt;
- Pen-4: De $GND$ pen. Dit is de grond-pen van de DHT11-sensor.

De datasheet raadt aan om een $5k\Omega$ *pull-up resistor* te gebruiken wanneer de kabellengte langer is dan 20 meter.

Als tweede stap is ook een literatuuronderzoek gedaan naar de werking van de DHT11-sensor. Uit deze literatuuronderzoek [11] is gebleken dat de DHT11-sensor een *slave device* is. De DHT11-sensor stuurt pas gegevens wanneer de sensor wordt aangespoord door zijn *master device*. De MCU (in dit geval de *master*) doet een verzoek op de databus en wacht tot dat de sensor reageert met meetdata. De DHT11-sensor zal dan vervolgens de gemeten data terug koppelen aan de MCU. Voor het uitlezen van data door de MCU zal deze wijze toegepast moeten worden bij het programmeren. Voor het programmeren kunnen C of C++ als taal gebruikt worden.

➢ Onderzoek van de technische specificaties van de ESP32 MCU

De datasheet [12] van de ESP32 MCU is onderzocht. In Figuur 3 is de *pinout*-schema van deze MCU weergegeven. Hieronder is een lijst weergegeven met de essentiële technische specificaties die relevant zijn voor het beantwoorden van deze deelvraag.

- Wi-Fi: IEEE 802.11b/g/n, hierbij wordt alleen 2.4 GHz ondersteund;
- Dual-core 32-bit LX6 microprocessor;
- 448 KB ROM en 520 KB SRAM;
- 34 GPIO (*General Purpose Input Output*) pennen;
- Arduino (C++) *compatible*.

Als tweede stap is ook literatuuronderzoek [13] gedaan naar de werking van de ESP32 MCU. Deze MCU draait een *realtime* operating system genaamd FreeRTOS. Het is een *dual-core* systeem dat ontworpen is om de volgende protocollen te kunnen gebruiken:

- TCP/IP;
- 802.11 WLAN;
- MAC.

Deze MCU heeft 34 GPIO-pennen, en met deze pennen kunnen *slave*- en/of *master devices* op aangesloten worden. De ESP32 wordt geprogrammeerd in C en in C++ (Arduino). Om de ESP32 te programmeren in C++ zijn er ook Arduino bibliotheken beschikbaar.

➢ Literatuuronderzoek naar het uitlezen van meetgegevens door een MCU

Om de meetgegevens van de DHT11-sensor af te lezen, wordt gebruik gemaakt van programmeertaal C++, hierbij wordt de `<DHT11.h>` Arduino bibliotheek gebruikt [14]. De pennen dienen op de juiste manier aangesloten te worden, zoals eerder is aangegeven. Zie Figuur 1 voor de aansluitingsschema en Figuur 2 voor de schematische weergave ervan.

Nadat in het C++ code de `<DHT11.h>` Arduino bibliotheek is toegevoegd kan het DHT11-object worden aangemaakt. Tijdens het definiëren van het object moeten in de parameters aangegeven worden op welke GPIO-pen de sensor aangesloten is, en welke type DHT11-sensor het is, zie hieronder:

```
DHT11 dht_sensor(21, DHT11);
```

Om de data van de DHT11-sensor op een juiste manier af te kunnen lezen dienen variabelen aangemaakt te worden voor de luchtvochtigheid en temperatuur. Voor het opslaan van de gemeten data in deze variabelen, worden deze als volgt toegewezen:

```
float luchtVocht  = dht_sensor.readHumidity();
float tempGC = dht_sensor.readTemperature();
```

Om de juiste werking van DHT11-sensor te kunnen testen kan er eventueel gebruik gemaakt worden van de print functie [15], zie hieronder:

```
Serial.print("Luchtvochtigheid: ");
Serial.print(luchtVocht);
Serial.print("% RH");
Serial.print(" en ");
Serial.print("Temperatuur: ");
Serial.print(tempGC);
Serial.print("°C");
```

*B. Verken op welke elementaire wijze een Wi-Fi netwerk opgezet kan worden met de ESP32 MCU?*

➢ Bestuderen van ESP-IDF documentatie ter verkennen van Wi-Fi mogelijkheden

Om erachter te komen hoe op een simpele wijze een Wi-Fi netwerk te programmeren is met de ESP32 MCU, is eerst een documentatieonderzoek gedaan (*deskresearch*) naar welke manieren er beschikbaar zijn om Wi-Fi te configureren op de ESP32 MCU. Volgens de officiële *ESPressif-IDF documentation* [16] zijn er in totaal twee methoden beschikbaar om een Wi-Fi netwerk te configureren op de ESP32 MCU, te weten:

1) De *Access Point*-modus: Hiermee kan een AP (*access point*) geconfigureerd worden, waarna andere apparaten (die Wi-Fi ondersteunen) kunnen verbinden met de ESP32 MCU;
2) STA/AP-modus: Hiermee kan de ESP32 MCU zowel een client zijn van een andere AP, als een eigen AP terggelijk zijn.

Daarnaast zijn er ook nog andere configuratie mogelijkheden beschikbaar voor de AP, STA en STA/AP-modi, namelijk:

1) Verschillende beveiligingsmodi: denk aan beveiligingen zoals WPA, WPA2, WEP;
2) Continu-scan modus: De ESP32 MCU zal op een actieve wijze scannen naar toegangspunten;
3) Promiscue modus: Deze modus wordt gebruikt voor het monitoren en bijhouden van IEEE 802.11 (WLAN) pakketten.

➢ Vergelijk de verschillende WiFi methoden en configuratie mogelijkheden van ESP32 MCU om daarna een juiste keuze te kunnen maken voor een elementaire Wi-Fi netwerk opzet

Een elementaire Wi-Fi netwerk moet een los Wi-FI netwerk zijn, daarom valt de STA/AP af. Daarnaast moet dit netwerk simpel zijn en zijn de continu-scan- en promiscue modi niet van belang. Tenslotte is er gekozen om geen beveiligingsmodi te configureren vanwege elementaire opzet voor een Wi-FI netwerk. De aangegeven vergelijking heeft geresulteerd tot het toepassen van de AP methode, zonder toevoeging van extra configuratie mogelijkheden.

➢ Realiseer met de gekozen Wi-Fi netwerk mogelijkheid de benodigde elementaire Wi-Fi netwerk

Het configureren van de AP op de ESP32 MCU wordt gerealiseerd door de “WiFi.h” bibliotheek te gebruiken. In deze bibliotheek wordt de AP functionaliteit ook wel *SoftAP* genoemd. Daarnaast moet de AP een SSID (*Service Set IDentifier*, A.K.A. netwerk naam) hebben en kan dit met behulp van de volgende code toegepast worden [17]:

```
#include <WiFi.h>
/* Naam van netwerk: */
const char* ssid = "DHT11-sensor-netwerk";
/* (Geen) Wachtwoord van netwerk: */
const char* password = NULL;
```

De AP moet ook gekoppeld worden aan een IP-adres, zodat een gebruiker verbinding kan maken met de AP. Met deze onderstaande configuratie wordt het IP-adres, de default gateway en het subnet-masker van de AP geconfigureerd [18]:

```
IPAddress local_ip(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
```

Om nu de AP te starten hoeft alleen nog in de `void setup()` functie de `WiFi.SoftAP()` en `WiFi.softAPConfig` methoden opgeroepen te worden:

```
void setup() {
    Serial.begin(115200);
    WiFi.softAP(ssid, password);
    WiFi.softAPConfig(local_ip, gateway, subnet);
  }
```

*C. Op welke manier kunnen de meetgegevens continu ververst en gepresenteerd worden via de webinterface?*

➢ Literatuuronderzoek naar mogelijke programmeermogelijkheden die gebruikt kunnen worden om een webpagina te verversen

Volgens de uitgevoerde literatuuronderzoek [19] zijn er meerdere mogelijkheden om een webpagina te verversen. Een van de mogelijkheden is het gebruik van JS (*JavaScript*) code. De JS code gebruikt de `location.refresh()` methode om de webpagina te verversen. De JS code kan worden geïmplementeerd in de *source code* van de HTML webpagina.

Een andere mogelijkheid om de webpagina te verversen is het gebruik van `<meta>` binnen in de HTML code. De `<meta>` tag staat in de header van de HTML code en kan gemanipuleerd worden als *refresh* mogelijkheid [20]. Het nadeel van `<meta>` is dat gebruikt kan worden door spammers om zoekmachines voor de gek te houden [21].

Ook is het mogelijk om AJAX (*Asynchronous JavaScript And XML*) te gebruiken, alleen is dit complexer dan andere twee mogelijkheden, aangezien het een combinatie van webtechnologieën betreft [22].

➢ Op welke wijze kan de gekozen programmeermogelijkheid geïmplementeerd worden en welke functies of methoden horen daar bij

Om deze deelvraag te beantwoorden is er gekozen om *JavaScript* (JS) te gebruiken. De andere twee mogelijkheden AJAX en `<meta>` vallen af, omdat AJAX te ingewikkeld is voor dit simpele concept en `<meta>` onveiliger is dan JS in verband met spammers.

Uit het onderzoek is het duidelijk geworden dat met JS mogelijk is om een de webinterface continu *up-to-date* te houden. In JS wordt deze methode ook wel `location.reload()` genoemd. Als een *reload* wordt gebruikt in een webserver, is het wel essentieel om een delay toe te voegen, want anders zal de webserver te veel *refresh requests* krijgen. Een delay van 2 seconden is dan ook nodig voor de DHT11-sensor, aangezien de sensor minimaal 2 seconden nodig heeft om de data op een nauwkeurige manier te meten en beschikbaar te stellen.

Dit kan uitgevoerd worden in JS met de volgende code [23]:

```
// De webpagina om de 2 seconden verversen:
setTimeout(function(){
    location.reload();
}, 2000); // 2000 millisecondes = 2 secondes
```

➢ Onderzoek hoe en welke taal geschikt is om de realtime data te presenteren via de webinterface

Uit literatuuronderzoek [24] is gebleken dat er maar één gestandaardiseerde taal wordt gebruikt voor webinterfaces en dat is HTML (*Hypertext Markup Language*). HTML is de kern van een website en daar om heen kunnen andere talen gebruikt worden om de website een betere *user experience* aan te bieden. Voorbeelden van die talen kunnen zijn: CSS (*Cascading Style Sheets*), JS (*JavaScript*), PHP (*Hypertext Preprocessor*) of eventueel Python.

Uit een ander onderzoek [25] is een voorbeeld (HTML) code interessant gebleken om deze aandachtig te bestuderen, zodat het geleerde toegepast kan worden in dit onderzoek. Door het geleerde is een manier gevonden om de gemeten data *realtime* te presenteren (door gebruik van HTML/CSS/JS):

```
<!DOCTYPE html> <html>
 <head><meta charset=\"UTF-8\"><meta name=\"viewport
    \" content=\"width=device-width, initial-scale
    =1.0, user-scalable=no\">
  <title>realtime DHT11-sensor gegevens</title>
  <style>html { font-family: Helvetica; display:
    inline-block; margin: 0px auto; text-align:
    center;}
  h1 {color: #444444;margin: 50px auto 30px;}
  p {font-size: 24px;color: #888;}
  </style>
</head>
 <body>
  <h1>realtime DHT11-sensor gegevens:</h1>
   <p>Luchtvochtigheid: " + String(luchtVocht) + "%<
    /p>
   <p>Temperatuur: " + String(tempGC) + "C""</p>
   <script>
    setTimeout(function() {
     location.reload();
    }, 1000);
   </script>
 </body>
</html>
```

## V. Conclusies

De hoofdvraag "*Hoe kan een werkend systeem ontwikkeld worden met een MCU die de realtime gemeten luchtvochtigheid en temperatuurgegevens door de DHT-sensor kan uitlezen, verwerken en met een eigen Wi-Fi netwerk deze gegevens op een toegankelijke wijze kan presenteren via een webinterface op diverse end-devices?*" is opgedeeld in drie deelvragen voor het uit te voeren van het onderzoek om zo te komen tot een succesvolle resultaat.

### *A. Op welke simpele manier kunnen de meetgegevens van de DHT11-sensor realtime uitgelezen worden door een ESP32 MCU?*

Deze deelvraag omvat een hardware opzet en communicatie software om zo op een simpele manier de meetgegevens van DHT11-sensor realtime uit te kunnen lezen door de ESP32 MCU. De toegepaste onderzoeksmethode en de doorlopen stappen hebben geleid tot een succesvolle resultaat. De DHT11-sensor wordt middels seriële communicatie uitgelezen met de opgezette hardware, zie Figuur 1. Door het test functie toe te voegen heeft dat geholpen bij het controleren van het deelresultaat.

### *B. Verken op welke elementaire wijze een Wi-Fi netwerk opgezet kan worden met de ESP32 MCU?*

Uit de verkenning van de documentatieonderzoek van de ESP-IDF is gebleken dat de ESP32 MCU uitgebreide mogelijkheden heeft om een Wi-fi netwerk op te kunnen zetten. Echter, de vraag gaat over het opzetten van een elementaire Wi-Fi-netwerk. De tweede was van belang stap om te komen tot een juiste keuze voor een elementaire opzet van de Wi-Fi netwerk. Met de tweede stap en derde stap is er eerst een succesvolle selectie uit de standaard software bibliotheek gemaakt voor de AP modus en met de derde stap ook de Wi-Fi netwerk gerealiseerd is.

### C. Op welke wijze kunnen de meetgegevens continu ververst en gepresenteerd worden via een webinterface?

De literatuuronderzoek heeft geleid tot goede inzichten in verschillende oplossingsmogelijkheden. Daarnaast zijn de voor- en nadelen van het toepassen van de verschillende oplossingsmogelijkheden achterhaald om een juiste keuze te kunnen maken voor de opzet van een webinterface. Naast de literatuuronderzoek is er in afwijking op de stappenplan ook de tutorials geraadpleegd om voorbeeld codes te vinden en eventueel te gebruiken bij het implementeren van een oplossing. De gekozen methode met bijbehorende stappenplan en de aanvullende acties hebben geresulteerd in een efficiënte en effectieve oplossing voor deze deelvraag.

### D. Eindconclusie

Doordat alle drie de deelvragen succesvol zijn doorlopen en deze geïntegreerd beschouwd ook tot de gewenste resultaat heeft geleid, kan er geconcludeerd worden dat de uitvoering van dit onderzoek geslaagd is. Als eindproduct is er ook een werkend systeem ontwikkeld met een ESP32 MCU die de realtime gemeten luchtvochtigheid en temperatuurgegevens door de DHT11-sensor uitleest en verwerkt en daarna met een eigen Wi-Fi netwerk op een toegankelijke wijze de meetgegevens presenteert via een webinterface op diverse *end-devices*.

## VI. Discussie

In de discussie wordt gereflecteerd naar de werkwijze van hoe het hard- en software ontwerp heeft geleid tot een werkend systeem. Hierbij worden de geslaagde punten en mogelijke verbeterpunten van het onderzoek besproken.

In dit onderzoek wordt de ESP32 MCU ingezet om een AP te creëren, waarna andere gebruikers verbinding kunnen maken met die AP, zodat de webinterface bezocht kan worden. Deze AP is op een elementaire manier opgebouwd, omdat andere configuratieopties niet van belang waren op de probleemstelling. Echter had er wel beveiligingsmodi geconfigureerd kunnen worden, zodat het netwerk minimaal beschermd is tegen ongeautoriseerde gebruikers.

Een ander belangrijk verbeterpunt van het onderzochte ontwikkelde systeem is de toegankelijkheid van de meetgegevens via een webinterface op diverse *end-devices*. Hoewel de webinterface minimaal functioneel was en de *realtime* meetgegevens kon presenteren, had het ook uitgebreider gekund. Denk hierbij aan het bijhouden van de meetgegevens en deze te presenteren in een grafiek of het implementeren van waarschuwingssystemen, zodat gebruikers direct op de hoogte worden gesteld van eventuele waarschuwingsmetingen.

Een van de geslaagde punten is hoe het onderzoek uitgevoerd is. Het onderzoeken van andere artikelen heeft een breder inzicht gegeven op het domein van de DHT11-sensor, microcontrollers en het opzetten van een webinterface.

## VII. Aanbevelingen

Hoewel het onderzoek succesvol was in het ontwikkelen van een werkend systeem met een ESP32 MCU voor het uitlezen en verwerken van realtime gemeten luchtvochtigheids- en temperatuurgegevens via de DHT11-sensor, zijn er enkele aanbevelingen die kunnen helpen bij het uitbreiden van het systeem.
Het uitbreiden van dit concept is zeker mogelijk en aan te bevelen. Aanbevolen uitbreidpunten kunnen zijn:

- Configuratie opties van de AP: In dit onderzoek was het essentieel om een elementaire Wi-Fi netwerk te configureren, zodat gebruikers op afstand de webinterface kunnen bezoeken. Daarom was gekozen om geen promiscuemodus, continu-scan-modus en beveiligingsmodi te configureren voor de AP. In een vervolgonderzoek zou eventueel beveiligingsmodi geïmplementeerd kunnen worden, zodat alleen toegankelijke gebruikers verbinding kunnen maken met de AP die toegang hebben de webinterface kunnen bezoeken.

- AP uitbreiden met STA/AP-modus: door STA/AP-modus te configureren in plaats van alleen AP kan de ESP32 MCU zowel een client zijn van een thuisnetwerk als een eigen AP zijn. Gebruikers die thuis wonen hoeven dan alleen het IP-adres van de ESP32 in te vullen om de webinterface te bezoeken. Andere ongeautoriseerde mensen (zoals visite) kunnen verbinden met de AP en daarvandaan de webinterface bezoeken. Echter is het belangrijk dat een juiste beveiliging wordt ingesteld.

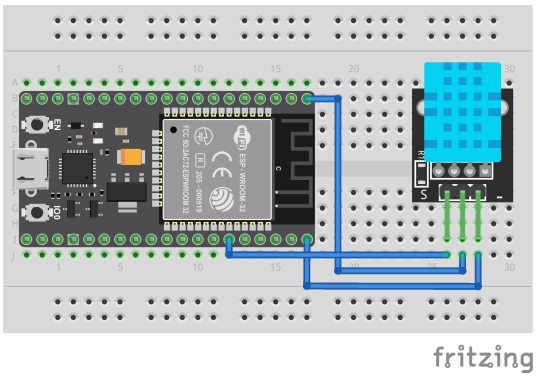
- Het hardware-matige uitbreiden met een LCD of OLED display: Ook is deze uitbreiding aan te raden, want met een scherm is het mogelijk om de *realtime* data te presenteren. Een pluspunt van deze uitbreiding zou zijn dat de display gemonteerd kan worden aan een muur, waarna de gemeten data direct leesbaar is.

- Database: Naast de realtime gemeten data te presenteren via de webinterface, zou het helemaal mooi zijn als de gegevens worden opgeslagen in een database. Door een database te gebruiken kunnen gegevens uitgezet worden tegen de tijd, waarna een grafiek op te stellen is.

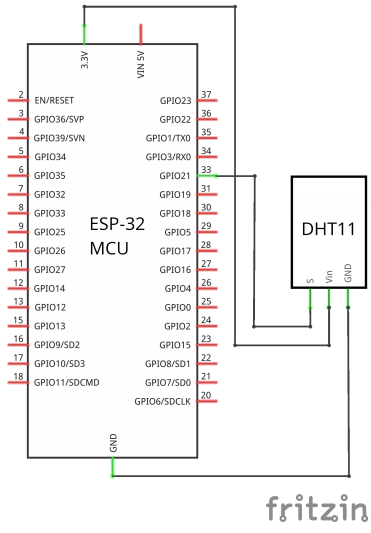

## Dankbetuigingen

De auteur wil zijn dank uitspreken aan Sander Gieling en Seethu Christopher voor hun ondersteuning en begeleiding voor het schrijven van dit artikel. Doordat S. Gieling en S. Christopher hun kennis hebben gedeeld over het schrijven van een onderzoeksartikel tijdens de lessen, heeft de auteur een beter inzicht gekregen over het schrijven van een artikel.

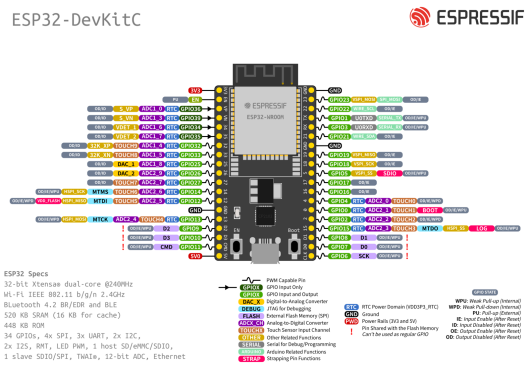
## BIJLAGEN




Figuur 1. Aansluitingsschema tussen DHT11-sensor en ESP32 MCU.




Figuur 2. Schematische weergave (CAD-tekening) tussen DHT11-sensor en ESP32 MCU




Figuur 3. *Pinout*-schema van ESP32 microcontroller (van datasheet).

## REFERENTIES


[1] B. Vallabh, A. Khan, D. Nandan, and M. Choubisa, "Data acquisition technique for temperature measurement through dht11 sensor," in *Proceedings of Second International Conference on Smart Energy and Communication*, D. Goyal, P. Chaturvedi, A. K. Nagar, and S. Purohit, Eds. Singapore: Springer Singapore, 2021, pp. 547–555.

[2] I. Krishna and K. Lavanya, "Intelligent home automation system using bitvoicer," in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, pp. 14–20. [Online]. Available: https://doi.org/10.1109/ISCO.2017.7855973

[3] P. Suanpang and P. Jamjuntr, "A smart farm prototype with an internet of things (iot) case study: Thailand," *Journal of Advanced Agricultural Technologies*, vol. 6, pp. 241–245, 01 2019. [Online]. Available: https://doi.org/10.18178/joaat.6.4.241-245

[4] M. S. Novelan and M. Amin, "Monitoring system for temperature and humidity measurements with dht11 sensor using nodemcu," *International Journal of Innovative Science and Research Technology*, vol. 5, no. 10, pp. 123–128, 2020.

[5] D. Srivastava, A. Kesarwani, and S. Dubey, "Measurement of temperature and humidity by using arduino tool and dht11," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 12, pp. 876–878, 2018.

[6] J. Patrick, "Serial protocols compared," *Embedded Systems Programming*, vol. 19, 2002. [Online]. Available: http://coecsl.ece.illinois.edu/se423/datasheets/SerialProtocolsCompared.pdf

[7] G. Mishra and D. Kaavya, "Interfacing atmospheric variables to web interface using arduino," in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICAMMAET.2017.8186709

[8] P. Macheso, S. Chisale, C. Daka, N. Dzupire, J. Mlatho, and D. Mukanyirigira, "Design of standalone asynchronous esp32 webserver for temperature and humidity monitoring," in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2021, pp. 635–638. [Online]. Available: https://doi.org/10.1109/ICACCS51430.2021.9441845

[9] S. Santos, "Esp32 vs esp8266 pros and cons," https://makeradvisor.com/esp32-vs-esp8266/, September 2021, [Accessed 2023-06-13].

[10] *DHT11 Humidity and Temperature Sensor*, Mouser, 3 2018. [Online]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[11] W. Gay, *DHT11 Sensor*. Berkeley, CA: Apress, 2018, pp. 399–418. [Online]. Available: https://doi.org/10.1007/978-1-4842-3948-3_22

[12] *ESP32 Series Datasheet*, Espressif Systems, 1 2023, rev. 4. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[13] A. Maier, A. Sharp, and Y. Vagapov, *Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things*, 2017. [Online]. Available: https://doi.org/10.1109/ITECHA.2017.8101926

[14] N. Cameron, *Arduino Applied*. Apress, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4842-3960-5

[15] Y. A. Badamasi, "The working principle of an arduino," in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, 2014, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ICECCO.2014.6997578

[16] "ESP-IDF Programming Documentation ESP32 ESP-IDF Programming Guide latest documentation," June 2023, release v5.2-dev-1051-g17d6768e65. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/esp-idf-en-v5.2-dev-1051-g17d6768e65-esp32.pdf

[17] W. Amriou, "Esp32-wifi-access-point," https://gist.github.com/walidamriou/fa5131e80b457fed488249a366a9bad3, 2021, [Accessed 2023-06-13].

[18] A. Upesy, "Create a wifi access point with an esp32," https://www.upesy.com/blogs/tutorials/how-create-a-wifi-acces-point-with-esp32, January 2023, [Accessed 2023-06-13].

[19] P. T. Lau, "Event-based remote attacks in html5-based mobile apps," in *Computer Security*, A. P. Fournaris, M. Athanatos, K. Lampropoulos, S. Ioannidis, G. Hatzivasilis, E. Damiani, H. Abie, S. Ranise, L. Verderame, A. Siena, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2020, pp. 49–63.

[20] B. Lindgren, "Measurement techniques on distance using a web browser," *International Journal of Innovation in Science and Mathematics Education*, vol. 7, no. 1, 2001.

[21] J. Kyrnin, "s," https://www.thoughtco.com/meta-refresh-tag-3469046, March 2020, [Accessed 2023-06-14].

[22] A. K. Singh, "Ajax asynchronous database refresh," *International Journal of Information and Communication Technology*, vol. 2, no. 8, 2012.

[23] J. Olawanle, "Method 5: Refresh a page using location.reload with a delay," https://www.freecodecamp.org/news/javascript-refresh-page-how-to-reload-a-page-in-js/, April 2023, [Accessed 2023-06-13].

[24] J. Krause, *HTML: Hypertext Markup Language*. Berkeley, CA: Apress, 2016, pp. 39–63. [Online]. Available: https://doi.org/10.1007/978-1-4842-2499-1_3

[25] R. Teja, "How to create an esp32 web server," https://www.electronicshub.org/esp32-web-server/, March 2021, [Accessed 2023-06-13].