



Realisatieverslag

Felix Hoebrechts

IOT 2021

Stageperiode 20/09 – 17/12



Inhoudsopgave

1	Inleiding.....	5
2	Opdracht	6
3	Kennismaking met de werkomgeving.....	7
4	De gebruikte onderdelen	8
4.1	Meter M23 112-10M	8
4.2	ESP32.....	8
4.3	USB-adapter	9
5	Realisatie.....	10
5.1	Aansluiting	10
5.1.1	Modbus aansluiting met energiemeter	10
5.1.2	Modbus aansluiting met USB-adapter en energiemeter	14
5.1.3	Modbus aansluiting met ESP32 en energiemeter	15
5.1.4	Modbus aansluiting met ESP32, USB-adapter en energiemeter	15
5.2	Modbus	16
5.2.1	USB-adapter niet detecteren	16
5.2.2	Manueel sturen.....	17
5.2.3	Software Sturen	20
5.3	Website	24
5.3.1	API	25
5.3.2	Website	25
5.4	MQTT.....	27
5.4.1	MQTT broker	27
5.4.2	MQTT explorer	27
5.4.3	MQTT berichten versturen.....	28
5.4.4	MQTT berichten ontvangen	28
5.5	mDNS	29
5.6	Start afgewerkt product.....	29
6	Besluit.....	30
7	Bronnen.....	31
7.1	Modbus info	31
7.2	ESP32 Libraries.....	31
7.2.1	Modbus	31
7.2.2	HTTP server	31
7.2.3	MQTT.....	31

7.2.4	mDNS	31
-------	------------	----

1 Inleiding

Via de lijst van stages die we vanuit de school aangeboden kregen, ben ik op Bepowered uitgekomen. Het bedrijf is interessant omdat het werkt aan energiebeheer. Energie vormt de basis van ons leven. We hebben natuurlijke hulpbronnen zoals zon, water en wind die zorgen voor ons maximaal comfort. Om een minimale impact te hebben op het milieu ontwikkelt Bepowered projecten om het elektriciteitsnet te monitoren en te beheren.

Ze bieden gebruiksvriendelijke oplossingen om tot een gewenst economisch en ecologisch hoogtepunt te komen. Ze bouwen ecosystemen voor energiebeheer om de lokale energieproductie, -opslag en -vraag te controleren en in evenwicht te houden.

Projecten waar ik me in mijn vrije tijd ook mee bezig hou, hebben hier aansluiting mee. Het is een zeer interessant, actueel onderwerp.

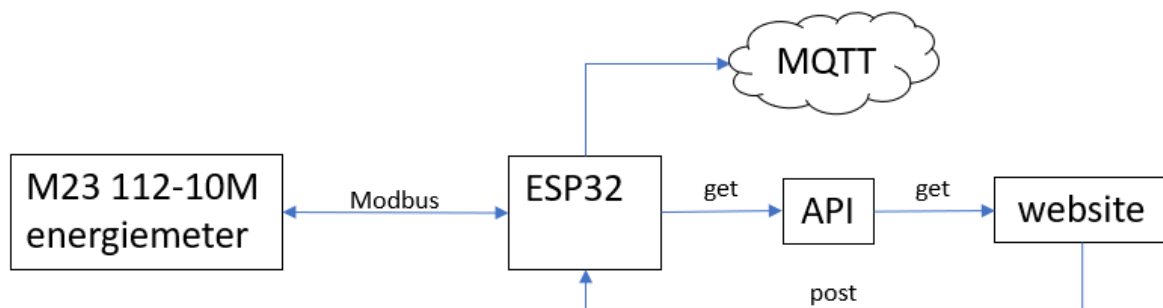
Ook door mijn ervaring met raspberry pi's en arduino's heb ik al kennis opgedaan van verschillende programmeertalen zoals python, javascript, C. Ik heb ook gebruik gemaakt van mijn html en css kennis om deze projecten een betere layout te geven.

Bepowered maakt gebruik van volgende protocollen: MQTT, HTTP API's, websockets, mDNS.

Het is een bedrijf met Bart Eestermans als founder en CEO, een embedded engineer, application developer en een web developer. De verslaggeving en de opdrachten gebeurden met de CEO en de embedded engineer.

2 Opdracht

Het verwachte resultaat van de opdracht is dat er een ESP geplaatst wordt op een plaats waar oplaadpalen voor elektrische voertuigen staan. De ESP zal een correcte verbinding maken door middel van het modbusprotocol met de energiemeters van de oplaadpalen. De opdracht bestond uit het uitlezen van waarden uit een energiemeter via modbus. Hierna moest er een embedded website aan gekoppeld worden om de data van de energiemeter op weer te geven. Vervolgens zal de ESP de nodige waarde inlezen en hiermee een json string opbouwen om op de API te posten. Indien de API met json correct opgebouwd is, zal de website zichzelf updaten en is er dus geen manuele handeling nodig om de data up-to-date te houden. Als alles correct aangesloten is, kan de embedded website een live beeld geven van de ingelezen waarde doordat een javascript de data van de API inleest. Daarna is ook MQTT toegevoegd om de data makkelijk in een correct formaat naar andere apparaten te versturen indien dit nodig zou zijn. Als laatste is ook mDNS toegevoegd om een zoektocht naar het IP-adres van de ESP32 te vermijden.



3 Kennismaking met de werkomgeving

Omdat ik eerder al gebruik gemaakt werd van Visual Studio Code, en extensies makkelijk toe te voegen zijn, zal deze stage gecodeerd worden in deze omgeving.

Voor er aan het project kon begonnen worden, moest eerst de juiste werkomgeving geïnstalleerd worden. Dit houdt in dat de juiste extensies toegevoegd worden aan de editor van Visual Studio code -. De gebruikte extensie is Espressif framework.

Vanuit het stagebedrijf is er een PCB ontworpen. Deze PCB is specifiek voor dit project en dient om modbusberichten en een internetverbinding te maken. Verder moesten de schema's die bij de speciaal ontworpen PCB horen, moesten grondig nagelezen worden zodat de juiste parameters konden gebruikt worden.

4 De gebruikte onderdelen

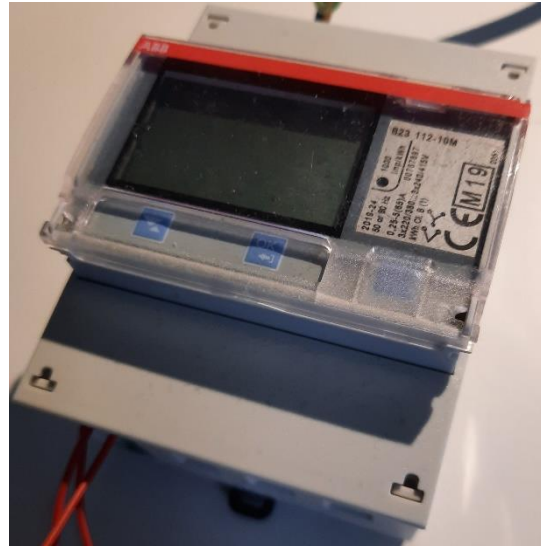
Hieronder worden de hardwarecomponenten besproken, onder andere een energiemeter, een ESP32 en een USB-adapter.

4.1 Meter M23 112-10M

Deze energiemeter wordt gebruikt om het verbruik van alle apparaten die aangesloten zijn achter de meter uit te meten.

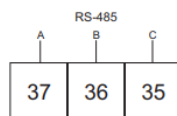
Natuurlijk zijn er een aantal onderdelen die verschillend zijn van meter tot meter. Dit kunnen verschillen zijn zoals het aansluiten van de meter tot het inlezen van data.

Op vraag van het bedrijf werd er gewerkt met modbus. Belangrijk om toe te voegen is dat M-bus niet hetzelfde is als modbus. In het onderstaande schema is ook duidelijk dat beiden op verschillende wijze aangesloten worden.

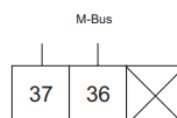


2.4.4 Communication

RS485



M-Bus



4.2 ESP32

De ESP32 is een micro-controller die gebruikt wordt voor het ontwikkelen van Internet Of Things toepassingen. Een micro-controller is een geïntegreerde schakeling met een microprocessor die wordt gebruikt om elektronische apparatuur te besturen.



4.3 USB-adapter

De USB-adapter werd gebruikt voor dataoverdracht en uploaden van data. Deze werd gebruikt in combinatie met het softwareprogramma "Qmodbus".



5 Realisatie

Het gerealiseerde project bestaat uit 6 delen. Het eerste deel start bij de energiemeter. Hierna verschoof de focus naar het bedraad modbus-communicatieprotocol. Dit is de manier waarop de meter communiceert met de ESP32. Na deze realisatie werd er een website toegevoegd waarop de waarde van de meter zichtbaar zijn. Het tweede communicatieprotocol is MQTT. Dit protocol maakt communicatie mogelijk over het internet. Het laatste deel dat gerealiseerd werd, is mDNS, een service die het IP-adres een duidelijke, herkenbare naam kan geven voor de gebruiker.

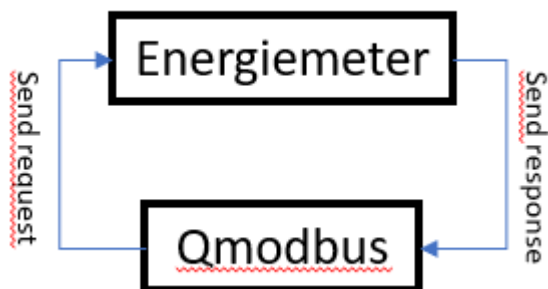
In onderstaande paragrafen worden deze delen toegelicht.

5.1 Aansluitingen

Het aansluiten van de ESP32 met de meter moet correct gebeuren. Hiervoor moest er onderzoek gedaan worden naar hoe modbus zijn aansluitingen verwacht tussen master en slave. In het onderzoek is gebruik gemaakt van onderstaande link om zowel de aansluiting als het modbusprotocol te studeren.

https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

Hieruit is gebleken dat modbus erg streng is met zijn aansluitingen. Als er één foute connectie is tussen de A en B aansluitingen kan dit het hele systeem stilleggen. Het aansluiten van de verschillende modbus apparaten hangt af van de huidige functie van het apparaat.



In dit geval werd er gebruik gemaakt van een USB-adapter om de verstuurd data van het protocol te monitoren. Deze adapter kan ook worden gebruikt om een enkel testbericht te versturen maar dan moest deze op een andere manier aangesloten worden om de werking van de ESP32 te behouden. Het tweede onderdeel is een speciaal gemaakte PCB die bestaat uit een ESP32 waar enkel de nodige pinnen uitgehaald zijn om het project uit te werken. Het derde en laatste onderdeel van deze stage was het hardware-onderdeel de energiemeter. Dit is een M23 112-10M energiemeter.

Zoals hierboven reeds beschreven is, is het correct aansluiten van de modbus connecties zeer belangrijk. 'Hieronder vind je aansluitschema's die in verschillende situaties werken.

5.1.1 Modbus aansluiting met energiemeter

Modbus kan data versturen en ontvangen. Aangezien dit project enkel data van de meter moet uitlezen, is de modbus functie code 3 voldoende en wordt hier dan ook gebruikt. In de

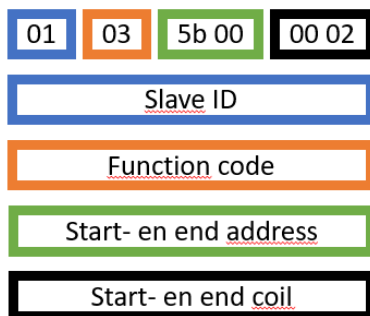
documentatie van de energiemeter staat aangegeven dat er 3 verschillende functiecodes gebruikt kunnen worden:

- Read Holding Registers (function code 3)
- Write Single Holding Registers (function code 6)
- Write Multiple Holding Registers (function code 16)

Modbus Function Codes Recognized by CSI Gateways

Function Code	Register Type
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

Modbusberichten zijn opgebouwd in verschillende onderdelen en elk onderdeel heeft zijn eigen doel.



Het bovenstaande commando (01 03 5b00 0002) kan verstuurd worden in het geval van deze energiemeter. Als alles correct verstuurd wordt en de errorcode (die automatisch toegevoegd wordt) correct is, kan er een antwoord terug verstuurd worden vanuit de meter. Als dit bericht verstuurd wordt naar deze meter, zal de waarde van de huidige spanning op L1 en N terug verstuurd worden. De gebruikte meter wordt gebruikt om 3-fases te meten. Deze meter bestaat uit 3 lijnen: L1, L2, L3 en een N die staat voor “neutral wire” of nuldraad.

Hieronder staat de uitleg en voorbeeld van een opgebouwd modbusbericht uit de datasheet van de meter.

General

Function code 3 is used to read measurement values or other information from the electricity meter. It is possible to read up to 125 consecutive registers at a time. This means that multiple values can be read in one request.

Request frame

A request frame has the following structure:

Slave Address	Function Code	Address	No. of Registers	Error Check
---------------	---------------	---------	------------------	-------------

Example of a request

The following is an example of a request. (read total energy import, etc...)

Slave address	0x01
Function code	0x03
Start address, high byte	0x50
Start address, low byte	0x00
No. of registers, high byte	0x00
No. of registers, low byte	0x18
Error check (CRC), high byte	0x54
Error check (CRC), low byte	0xC0

Response frame

A response frame has the following structure:

Slave Address	Function Code	Byte Count	Register Values	Error Check
---------------	---------------	------------	-----------------	-------------

Example of a response

The following is an example of a response:

Slave address	0x01
Function code	0x03
Byte count	0x30
Value of register 0x5000, high byte	0x00
Value of register 0x5000, low byte	0x15
...	
Value of register 0x5017, high byte	0xFF
Value of register 0x5017, low byte	0xFF
Error check (CRC), high byte	0XX
Error check (CRC), low byte	0XX

In this example, the slave with the Modbus address 1 responds to a read request. The number of data bytes is 0x30. The first register (0x5000) has the value 0x0015 and the last (0x5017) has the value 0xFFFF

In de onderstaande tabel staan alle registers die gebruikt kunnen worden bij functie code 3.
De "start reg" is dus de waarde die ingevuld moet worden bij het "start en het end address".

All registers in the following table are read only:

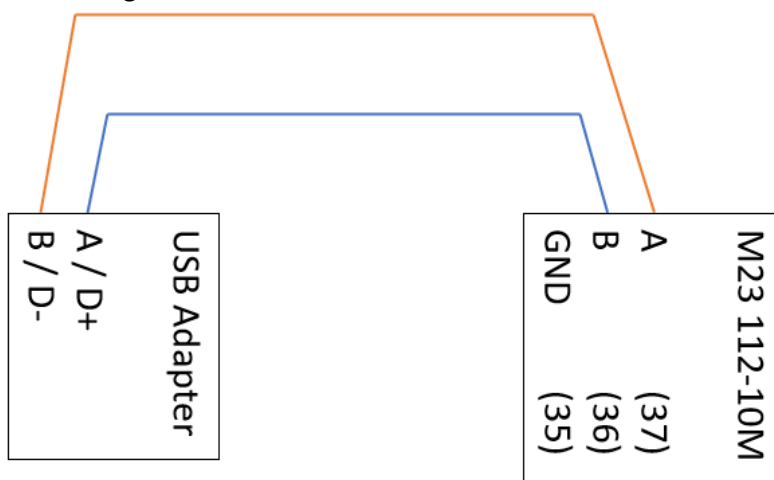
Quantity	Details	Start reg (Hex)	Size	Res.	Unit	Value range	Data type
Voltage	L1-N	5B00	2	0,1	V		Unsigned
Voltage	L2-N	5B02	2	0,1	V		Unsigned
Voltage	L3-N	5B04	2	0,1	V		Unsigned
Voltage	L1-L2	5B06	2	0,1	V		Unsigned
Voltage	L3-L2	5B08	2	0,1	V		Unsigned
Voltage	L1-L3	5B0A	2	0,1	V		Unsigned
Current	L1	5B0C	2	0,01	A		Unsigned
Current	L2	5B0E	2	0,01	A		Unsigned
Current	L3	5B10	2	0,01	A		Unsigned
Current	N	5B12	2	0,01	A		Unsigned
Active power	Total	5B14	2	0,01	W		Signed
Active power	L1	5B16	2	0,01	W		Signed
Active power	L2	5B18	2	0,01	W		Signed
Active power	L3	5B1A	2	0,01	W		Signed
Reactive power	Total	5B1C	2	0,01	var		Signed
Reactive power	L1	5B1E	2	0,01	var		Signed
Reactive power	L2	5B20	2	0,01	var		Signed
Reactive power	L3	5B22	2	0,01	var		Signed
Apparent power	Total	5B24	2	0,01	VA		Signed
Apparent power	L1	5B26	2	0,01	VA		Signed
Apparent power	L2	5B28	2	0,01	VA		Signed
Apparent power	L3	5B2A	2	0,01	VA		Signed
Frequency		5B2C	1	0,01	Hz		Unsigned
Phase angle power	Total	5B2D	1	0,1	°	-180° – +180°	Signed
Phase angle power	L1	5B2E	1	0,1	°	-180° – +180°	Signed
Phase angle power	L2	5B2F	1	0,1	°	-180° – +180°	Signed
Phase angle power	L3	5B30	1	0,1	°	-180° – +180°	Signed
Phase angle voltage	L1	5B31	1	0,1	°	-180° – +180°	Signed
Phase angle voltage	L2	5B32	1	0,1	°	-180° – +180°	Signed
Phase angle voltage	L3	5B33	1	0,1	°	-180° – +180°	Signed
Phase angle current	L1	5B37	1	0,1	°	-180° – +180°	Signed
Phase angle current	L2	5B38	1	0,1	°	-180° – +180°	Signed
Phase angle current	L3	5B39	1	0,1	°	-180° – +180°	Signed
Power factor	Total	5B3A	1	0,001	-	-1,000 – +1,000	Signed

5.1.2 Modbus aansluiting met USB-adapter en energiemeter

Deze aansluiting wordt gebruikt om met Qmodbus (een modbus monitoring service) individuele commando's door te sturen. Hierdoor zal de USB-adapter handelen als een master en zullen de antwoorden van de slave ook terug ingelezen worden. Als het verstuurde commando ingelezen wordt, dan zal er op Qmodbus ook een antwoord te zien zijn. Waar wel voor opgelet moet worden, is dat er ook foutieve commando's verstuurd kunnen worden. Hierop zal er dus geen antwoord worden teruggegeven.

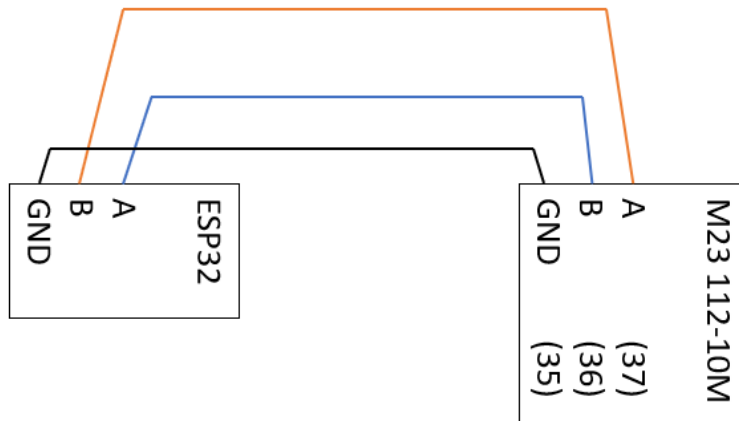
Met deze aansluiting wordt er enkel gebruik gemaakt van de USB-adapter en de energiemeter. Voor initiële testen is er nog geen gebruik gemaakt van de ESP32.

Het schema hieronder is hoe de USB-adapter en de meter aangesloten waren tijdens de initiële testen die gedaan werden met Qmodbus.



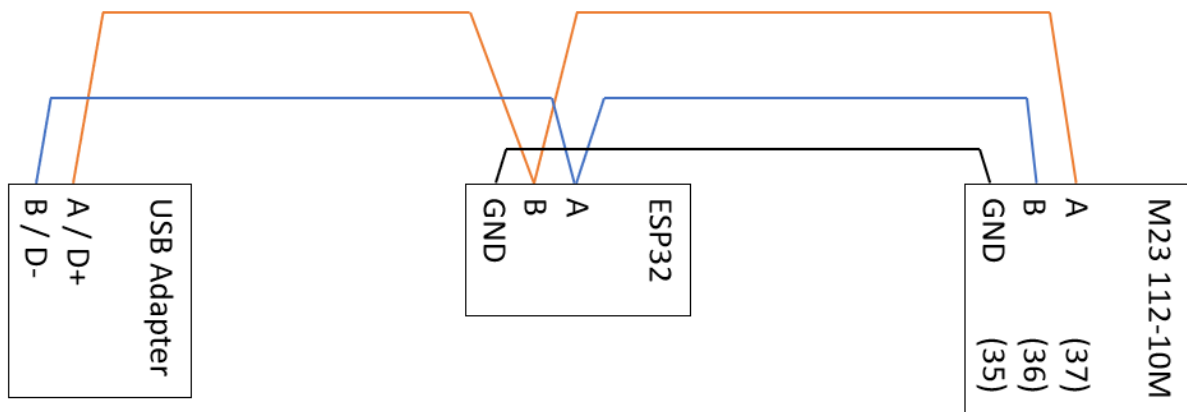
5.1.3 Modbus aansluiting met ESP32 en energiemeter

Om het uiteindelijke resultaat te behalen, is het de bedoeling dat er geen monitoring onderdelen meer in het project zitten. Dit wil dus zeggen dat het project zal eindigen met een ESP32 en een energiemeter.



5.1.4 Modbus aansluiting met ESP32, USB-adapter en energiemeter

Ook zijn er tests gedaan met het combineren van Qmodbus en de ESP32. De werking van de ESP32 was succesvol, de verwachte waarden werden ingelezen via de ESP32. De waarden die de Qmodbus kon inlezen, waren daarentegen niet de waarden die verwacht werden. Hier kwamen waarden terug die geen betekenis hadden.



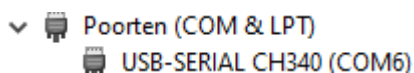
5.2 Modbus

Voordat de ESP32 toegevoegd werd in het project is er eerst onderzoek gedaan naar Modbus en de werking ervan. Er is gebruik gemaakt van Qmodbus. Qmodbus is een monitoring software die ook in staat is om enkele berichten te versturen. Om data zowel te kunnen uitlezen als versturen, is het gebruik van de USB-adapter nodig.

Eerst en vooral moet de USB-adapter correct aangesloten zijn met de energiemeter. In het hierboven beschreven hoofdstuk staat een [aansluitschema](#) waar de meter op de correcte manier aangesloten is met de USB-adapter. Indien deze correct aangesloten is en de USB-adapter verbonden is met een computer, moet er gecontroleerd worden of de adapter gedetecteerd wordt door de computer.

De beste manier om dit te doen is apparaatbeheer openen en zoeken tussen “USB-apparaten” of “Poorten”.

Zoals op onderstaande afbeelding te zien is, is deze correct verbonden. Een belangrijk detail is wel dat ook de kabel om de ESP32 te flashen hier zichtbaar zal zijn.



5.2.1 USB-adapter niet detecteren

Het is mogelijk dat de USB-adapter niet gedetecteerd wordt op een Windows OS. Ook is het mogelijk dat de adapter gedetecteerd wordt maar dat windows automatisch foutieve drivers toevoegd. Ondanks online op enkele forums aangegeven is, dat er geen drivers nodig zijn, zijn de juiste drivers wel degelijk nodig.

Als de foutieve drivers geïnstalleerd zijn kan de USB-adapter zichtbaar zijn. Het is mogelijk dat de naam van de adapter “USB2.0-Ser!” is. Deze kan dan gevonden worden onder “andere apparaten”. Dit is een resultaat van een persoonlijke ondervinding en kan verschillend zijn van andere resultaten. Dit kan een indicatie zijn van foutief geïnstalleerde drivers.

Met de onderstaande link kunnen de juiste drivers geïnstalleerd worden.

<https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all#drivers-if-you-need-them>

Download de “windows EXE” versie en deïnstalleer de foutieve drivers die gevonden kunnen worden onder de “andere apparaten/USB2.0-Ser!”. Vervolgens kan de nieuwe driver geïnstalleerd worden.

Na de installatie is het mogelijk dat de computer herstart moet worden voordat de adapter correct zichtbaar wordt. Na alle eerdere stappen te voltooien, zou de adapter correct zichtbaar moeten zijn.

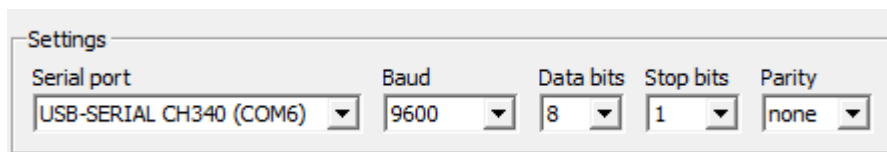
5.2.2 Manueel sturen

Qmodbus is monitoring software voor modbus. Met deze software is het ook mogelijk om berichten te versturen. Als de aansluitingen correct gedaan zijn zoals eerder weergegeven in [Modbus enkel USB en energiemeter](#), zou (met de juiste instellingen) Qmodbus correct werken. Er zijn enkele instellingen die Qmodbus nodig heeft om zowel berichten te lezen als uit te schrijven. Dit houdt in dat de juiste Serial port doorgegeven moet worden (indien de juiste poort niet zichtbaar is het mogelijk dat de [USB-adapter niet detecteerbaar](#) is).

Ook moet er opgelet worden op de connectie die de computer maakt met het apparaat. Als de USB-adapter uitgetrokken is, wanneer Qmodbus actief is, dan zal Qmodbus niet langer berichten versturen. De software zal dus opnieuw opgestart moeten worden voordat deze terug berichten kan versturen.

Afhankelijk van de waarde die de meter voor modbus gebruikt, kunnen deze verschillen van de voorbeelden die hier gegeven worden. In het geval van de eerder benoemde [meter](#) zijn de nodige instellingen voor Qmodbus:

- USB-adapter selecteren: USB-SERIAL CH340
- Baud-rate van de meter = 9600
- Data bits: 8
- Stop bits: 1
- Parity: None



The screenshot shows a 'Settings' window with five dropdown menus for serial port configuration. The 'Serial port' menu is set to 'USB-SERIAL CH340 (COM6)'. The 'Baud' menu is set to '9600'. The 'Data bits' menu is set to '8'. The 'Stop bits' menu is set to '1'. The 'Parity' menu is set to 'none'.

Serial port	Baud	Data bits	Stop bits	Parity
USB-SERIAL CH340 (COM6)	9600	8	1	none

Ondanks dat het voorbeeld rekening houdt met de instellingen van de specifieke meter, kunnen de instellingen van de meter nog verschillen. De instellingen van de meter en Qmodbus moeten overeenkomen om een correcte werking te behalen. Met deze instellingen zou het al mogelijk moeten zijn om de monitoring functie te gebruiken.

Om berichten te versturen zal er een tweede deel van de instellingen moeten ingevuld worden. Deze instellingen bestaan uit een standaard modbusbericht (slave ID, Functie code, start- & end-address, start- & end-coils).

Bij een initiële activatie van modbus is in normale omstandigheden de slave ID 1. Aangezien de bedoeling is om waarden van de meter uit te lezen, maken we gebruik van functiecode 3 (Read Holding Registers). Het nadeel aan Qmodbus is dat er geen hexadecimale waarden doorgegeven kunnen worden maar deze wel nodig zijn om de gewenste waarden terug te krijgen. Om de juiste, hexadecimale waarden te krijgen, zal er eerst een omrekening gedaan moeten worden. Ook zal er rekening gehouden moeten worden met het feit dat deze software zowel het begin- als eindadres combineert. Dit zal duidelijker worden aan de hand van de voorbeeldafbeelding. De laatste instelling die doorgegeven moet worden is de "number of coils". Dit is afhankelijk van welke waarde opgevraagd wordt met in de "start address", die eerder opgevraagd werd in de datasheet van de specifieke meter. Hier zal ook de hoeveelheid coils worden meegegeven.

Onderstaande afbeelding is opgedeeld in 2 delen, het blauwe deel bestaat uit aanpasbare instellingen. Deze instellingen kunnen dus gewijzigd worden. Het tweede deel, aangeduid door de rode kader, is het deel dat weergeeft welk bericht er verstuurd zal worden.

ModBus Request

Slave ID	Function code	Start address	Num of coils
1	Read Holding Registers (0x03)	23296	2

☐ Display hex data

01 03 5b 00 00 02

Send

Op de onderstaande afbeelding is te zien welke ingestelde waarde overeenkomen met de waarde die uiteindelijk verzonden kunnen worden.

ModBus Request

Slave ID	Function code	Start address	Num of coils
1	Read Holding Registers (0x03)	23296	2

☐ Display hex data

01 03 5b 00 00 02

Send

Zoals eerder aangegeven is het met de monitoring software Qmodbus niet mogelijk om hexadecimale waarden door te geven. Dit maakt het moeilijker om de waarden (vooral het adres) te vinden. Aangezien er maar 1 instelbaar adres is, zal dit uitkomen op grote waarden die meegegeven worden in het start address.

In het geval van dit voorbeeld wordt een bericht met Qmodbus naar de meter verstuurd dat de spanning tussen L1-N terugstuurt. De hexadecimale code is in dit geval 01 03 5b 00 00 02. Deze waarde (address en coil) zijn terug te vinden in de datasheet van de meter. Zoals hier te zien is, is het address 5B00 en de size/num of coils 2.

Ook is de resolutie van de terugkrijgende waarde 0.1. Dit wil zeggen dat de uiteindelijke waarde factor 10 groter zal zijn dan werkelijke waarde (de normale spanning van een huishelijk stopcontact is 220-230V, de waarde die de meter zal doorsturen zal 2200-2300v weergeven).

Quantity	Details	Start reg (Hex)	Size	Res.	Unit	Value range	Data type
Voltage	L1-N	5B00	2	0,1	V		Unsigned

Zoals op onderstaande foto te zien is, werden er enkele berichten verstuurd. Er werd een request gestuurd die de spanning tussen L1-N opvraagt op het apparaat met het modbus address 1. Dit request werd gedaan met functie code 3 en 2 coils zoals in de modbus datasheet van de meter staat.

01 03 5b 00 00 02

Send

Nadat er op “Send” geklikt wordt, zal er snel een reactie komen van de meter. Zoals op onderstaande afbeelding te zien is, werd er een “request” en “response” gestuurd en ontvangen vanuit de meter.

ModBus requests/responses:						
	I/O	Slave ID	Function code	Start address	Num of coils	CRC
1	Req >>	1	3	23296	2	0000
2	<< Resp	1	3	23296	1	3d82

Hierop volgt de waarde die de meter juist verstuurd. Hieruit is te zien dat de meter zijn waarde doorstuurt. In deze afbeelding is het antwoord van de meter 00 00 en 08 BD omdat het 2 coils waren die opgevraagd werden. De 08 BD is de spanning van L1-N. Als dit vertaald wordt naar decimale waarde zal dit uitkomen op 2237. Zoals eerder vernoemd, zal dit met factor 10 vergroot zijn. De juiste waarde is dus 223,7V.

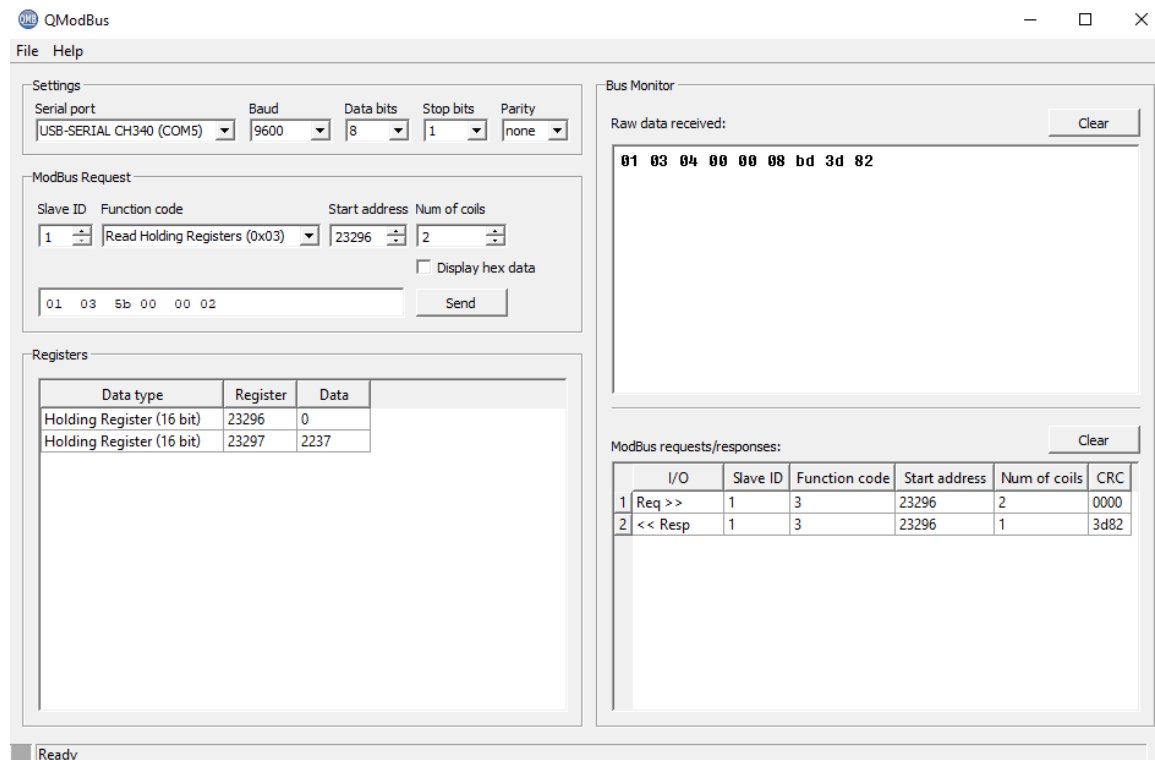
Raw data received:

01 03 04 00 00 08 bd 3d 82

In onderstaande afbeelding is te zien dat ook deze hexadecimale waarde omgerekend wordt naar de juiste decimale waarde.

Data type	Register	Data
Holding Register (16 bit)	23296	0
Holding Register (16 bit)	23297	2237

De laatste foto van Qmodbus is een volledige afbeelding waarop de locatie van alle onderdelen zichtbaar is.



5.2.3 Software Sturen

Het is natuurlijk niet de bedoeling dat de waarde elke keer manueel opgevraagd wordt. Dit neemt veel tijd in beslag en is bijna onmogelijk te monitoren. Om het continu monitoren mogelijk te maken, moet er een programma gebouwd worden.

5.2.3.1 ESP32 Pin layout

De PCB is opgebouwd met als hoofdcomponent een ESP32. Deze maakt gebruik van GPIO ofwel general input output pinnen. Enkele van deze GPIO pinnen op de ESP32 kunnen voor meer dan een binaire input of output gebruikt worden.

GPIO 9 => RXD1

GPIO 10 => TXD1

GPIO 14 => LED GREEN

GPIO 12 => LED GREEN

GPIO 33 => LED GREEN

GPIO 32 => BUTTON IN

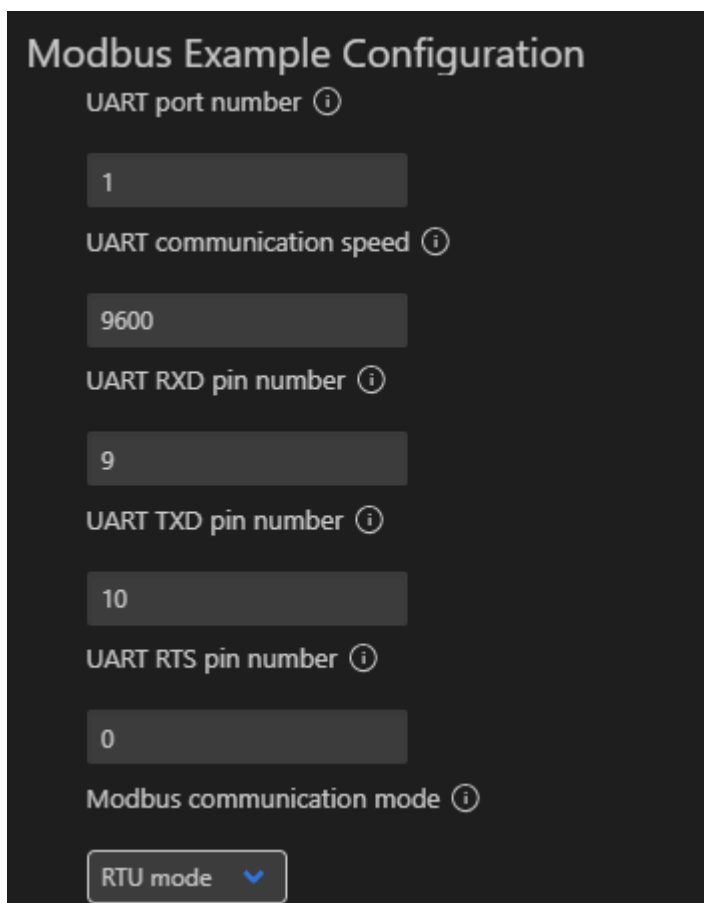
Deze gpio pinnen zijn belangrijk voor het sturen en ontvangen van modbusberichten.

5.2.3.2 *Guru meditation error core 0 panic'ed (storeprohibited). Exeption was unhandled*

Natuurlijk moeten er ergens de waarden van de pinnen, communicatiesnelheid en de UART-poortnummer doorgegeven worden. Indien één van deze waarden foutief ingevuld is, kan deze een error voorkomen.

Bij testing is dit veroorzaakt door de waarde van de “UART RTS pin number” in te vullen. Dit veld is niet verplicht in te vullen en dient dus leeg gelaten of naar “0” gezet te worden in het geval dat deze niet aangesloten is.

In het geval dat er geen aanpassingen gemaakt worden aan het bordje, is het mogelijk om onderstaande waarden te kopiëren. Dit zijn de waarden die in het configuratiebestand meegegeven worden. Ook belangrijk is om de communicatiesnelheid de juiste “baud rate” door te geven. Dit zal afhankelijk zijn van de snelheid die op de meter ingesteld is.



The image shows a configuration form titled "Modbus Example Configuration" with a dark background. It contains several input fields and a dropdown menu, each with an information icon (i) to its right. The fields are: "UART port number" with value "1", "UART communication speed" with value "9600", "UART RXD pin number" with value "9", "UART TXD pin number" with value "10", "UART RTS pin number" with value "0", and "Modbus communication mode" with a dropdown menu showing "RTU mode".

Configuration Parameter	Value
UART port number	1
UART communication speed	9600
UART RXD pin number	9
UART TXD pin number	10
UART RTS pin number	0
Modbus communication mode	RTU mode

Loadprohibited of storeprohibited is een error die kan ontstaan als er een verkeerde waarde wordt gegeven aan een variabele. Het is dus ook mogelijk dat de error niet opgelost kan worden door bovenstaande uitleg.

5.2.3.3 Modbus programma

Het modbusprogramma begint met een lijst die de modbusberichten gaat opbouwen. De waarden in deze lijst komen overeen met wat modbus verstuurd. Hier zijn ook wat extra waarden aan toegevoegd zoals een CID die later nog gebruikt wordt. Er zal ook een eenheid en een parameternaam toegevoegd worden.

Deze berichten worden dan correct geformatteerd en verstuurd via modbus. CRC ofwel errordetectie wordt ook hier net zoals bij het manueel sturen automatisch toegevoegd. Er zal dus geen extra waarde in de lijst toegevoegd moeten worden.

In dit geval worden er 7 verschillende waarden opgevraagd. De spanning tussen L1-N, L2-N en L3-N en het vermogen op L1, L2, L3 en het totaal vermogen.

Quantity	Details	Start reg (Hex)	Size	Res.	Unit	Value range	Data type
Voltage	L1-N	5B00	2	0,1	V		Unsigned
Voltage	L2-N	5B02	2	0,1	V		Unsigned
Voltage	L3-N	5B04	2	0,1	V		Unsigned
Active power	Total	5B14	2	0,01	W		Signed
Active power	L1	5B16	2	0,01	W		Signed
Active power	L2	5B18	2	0,01	W		Signed
Active power	L3	5B1A	2	0,01	W		Signed

Voor elk van deze onderdelen moet er ook op gelet worden dat de resolutie correct is. Dit is de nauwkeurigheid waarop de modbusberichten kunnen reageren. Aangezien de berichten met het antwoord geen komma's doorsturen, moet hier in het project later nog rekening mee gehouden moeten worden.

Een voorbeeld hiervan is, ondanks de waarden van een stopcontact rond de 220V tot 230V liggen, zal de teruggekregen waarde van de meter rond de 2200 tot 2300 zijn.

Het voordeel van de ESP32 is dat deze zeer snel opstart. Maar als deze op dezelfde moment opstart als de meter, is het mogelijk dat de meter nog niet correct opgestart is. Dit wil dus zeggen dat de modbus requests die de ESP32 naar de meter stuurt niet zullen verwerkt worden. Modbus zal hierdoor een time-out (0X107) error weergeven. Doordat een time-out error weergegeven wordt, zal er dus geen waarde correct ingelezen worden. De meter zal zijn requests blijven sturen zolang deze aan staat. Als er, nadat de meter correct opgestart is, waarden correct ingelezen worden, zal het project zijn normale werking verder zetten.

In het afgewerkte project is dit geen probleem omdat de opstartsequentie van de ESP32 lang genoeg duurt om de meter correct op te starten.

De CID's (Computer IDentificatie) worden gebruikt om ervoor te zorgen dat elke request correct uitgevoerd wordt. Dit wil dus zeggen dat deze uniek zijn. Hiervoor is er dan ook een "enum" functie gebruikt. Deze zorgt ervoor dat de waarden van de CID's optellend zijn van het eerst aangegeven getal. Indien deze niet uniek zijn, kan er ook een error weergegeven worden tijdens het monitoren van de software. Deze error is moeilijker te detecteren omdat de ESP32 dit ziet als een "fatal error". Dit wil zeggen dat de ESP32 zichzelf zal herstarten. Als deze lijst van CID's manueel aangemaakt of aangepast is, en deze lijst niet compleet is, zal het programma dit ook zien als een "fatal error". Ook in dit scenario zal de ESP32 zichzelf continu herstarten.

Het antwoord van een modbusbericht met 1 register is een 8-bit integer. Aangezien deze meter gebruik maakt van 2 registers zal er dus een uint16_t gebruikt moeten worden.

Het is mogelijk dat de ESP32 de waarde die de meter terugstuurt, verkeerd inleest. Hiermee wordt bedoeld dat CID 1 bijvoorbeeld de waarde 2100 verwacht, CID 2 de waarde 200 en CID 3 de waarde 200. De waarden die uiteindelijk ontvangen worden, zouden dan voor CID 1 = 200, CID 2 = 2100 en CID 3 = 200 zijn. Dit kan een probleem worden aangezien de verkeerde waarden worden toegekend aan de foutieve variabelen.

Het herstarten van zowel de meter als de ESP32 kan deze fout oplossen.

5.2.3.4 E (3280) MB_CONTROLLER_MASTER: mbc_master_get_parameter(83): Master get parameter failure, error=(0x107) (ESP_ERR_TIMEOUT).

E (3280) MASTER_TEST: Characteristic #0 (Data_channel_0) read fail, err = 0x107 (ESP_ERR_TIMEOUT).

Zoals eerder vermeld is het mogelijk dat deze error voorkomt tijdens het opstarten van zowel de meter als de ESP32. De kans bestaat dat deze error ook opduikt als de pinnen niet correct aangesloten zijn. In dit geval identificeert het programma niet welke pin correct is aangesloten. Alles moet manueel nagekeken worden.

5.3 Website

Nu data ingelezen kan worden, moet het ook op de juiste manier gepresenteerd worden. Dit wil dus zeggen dat er een website gemaakt moet worden.

```
static const httpd_uri_t hello = {
    .uri      = "/",
    .method   = HTTP_GET,
    .handler  = hello_get_handler,
    /* Let's pass response string in user
     * context to demonstrate it's usage */
    .user_ctx = server_root_cert_pem_start
};

static const httpd_uri_t api = {
    .uri      = "/api",
    .method   = HTTP_GET,
    .handler  = hello_get_handler,
    /* Let's pass response string in user
     * context to demonstrate it's usage */
    .user_ctx = GLOBAL_str
};
```

In de afbeelding hierboven is te zien hoe de waarden van de website opgebouwd zijn in de code.

De belangrijkste waarden die aangepast moeten worden zijn de .URI en de .user_ctx, de .URI is de link die meegegeven zal worden om de webpagina te bereiken. Dit wil niet zeggen dat dit de volledige link is maar eerder de extensie. In andere woorden, eerst komt het IP-adres met daarop volgend de URI. Op deze manier is dan de website bereikbaar.

De .user_ctx is de context of inhoud van de pagina. Hier kan een string in meegegeven worden. Deze string zal gebruikt worden voor de inhoud van de website. Voor de API is het mogelijk om de string op te bouwen in de C-code waar de data van de meters wordt ingelezen. Hierna moet zorgvuldig de syntax van json gebruikt worden. De juiste data moet later gegenereerd worden om deze op een goede wijze te kunnen gebruiken.

Het is mogelijk om een website op te bouwen in de C-code. Maar dit zal de code zeer onleesbaar maken. Hier moet voor een alternatief gezocht worden. Het toevoegen van een html pagina zou voor een leesbare code zorgen. Er moet gezorgd worden voor een connectie. Het html bestand zal moeten toegevoegd worden aan het project zodat dit bestand naar de esp geflashed kan worden. Om dit toe te voegen zal er in de Cmake files een `EMBED_FILES` of `EMBED_TXTFILES` toegevoegd worden.

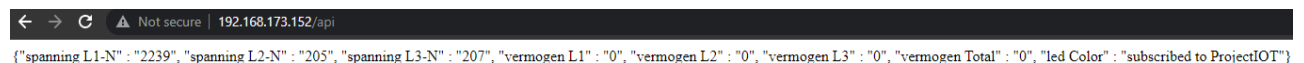
```
1  set(PROJECT_NAME "modbus_master")
2
3  ✓ idf_component_register(SRCS "master.c"
4                          INCLUDE_DIRS "."
5                          EMBED_TXTFILES index.html)
```


5.3.1 API

Hier wordt gebruik gemaakt van een string die zich op een pagina zelf aanpast aan de ingelezen data. Om de string correct te formatteren kan de C-functie “sprintf” gebruikt worden. Hierdoor kunnen meerdere formaten (int, str, hex, etc) in de C-code gebruikt worden om een enkele string op te bouwen met het juiste formaat (in dit geval json).

Eerst en vooral zal het modbus programma éénmalig doorlopen worden. Na 1 cycli zal de string correct geformatteerd worden met de waarde van de eerste metingen. Voordat de eerste metingen gedaan zijn, zal de string leeg zijn. Dit wil zeggen dat op dit korte moment de API leeg is. Zolang de API leeg is, zal de website N/A-waarde weergeven. Van het moment dat de API waarde heeft, dan zal de hoofdpagina waarden weergeven, zelfs als de waarde “0” zijn.

Hieronder is een afbeelding van de API.



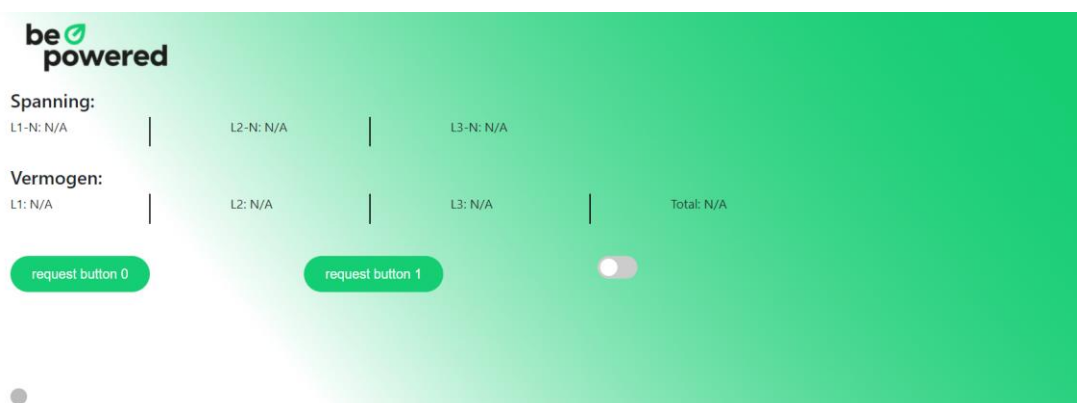
5.3.2 Website

Zoals eerder genoteerd zal de gebruikte website voor het monitoren opgebouwd zijn uit een html bestand. Deze zal gebruik maken van de API om de data te vernieuwen op de website. De data herladen zal gebeuren aan de hand van een javascript. Het javascript maakt gebruik van de API om zijn data te herladen. De webpagina hoeft niet herladen te worden om nieuwe data op de webpagina te zien. Het javascript haalt de data binnen en herlaadt die continu aan de hand van de data op de API.

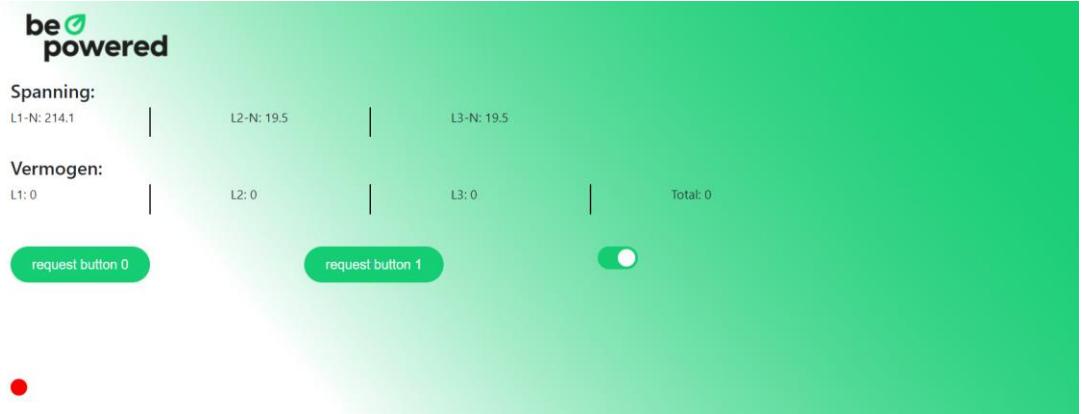
```
extern const uint8_t server_root_cert_pem_start[] asm("_binary_index_html_start");
```

Het bovenstaand commando haalt de html pagina binnen met de C-code. Bij het opstarten van de ESP32 start de code ook de webpagina. De webpagina staat nu in de variabele “server_root_cert_pem_start”. Deze is ook de gebruikte variabele in .user_ctx. Op deze manier kan de website gestart worden met de C-code.

Hieronder staat een afbeelding van de website die nog geen waarde ingelezen heeft van de API.



Hieronder staat een afbeelding van de website waar waarden uit de API gehaald zijn. De “toggle button” die hier ook actief is. Deze knop maakt ook de RGB-led rood op het bordje.



5.4 MQTT

MQTT is ook een deel van het project. De data die op de API geplaatst wordt, zal ook verstuurd worden in hetzelfde json formaat. Het aantal berichten zal moeten gelimiteerd worden om het netwerk niet te hard te belasten. Om dit op te lossen zal er elke keer dat de variabele van de API aangepast wordt, ook een bericht verstuurd worden. Dit bericht wordt opgebouwd nadat alle modbus waarden opgevraagd zijn. Aangezien er een vertraging nodig is tussen de polls, om deze correct in te kunnen lezen, is het, na het opbouwen van dit bericht na alle data in te lezen, de ideale moment om het MQTT bericht te versturen.

Ook moet de "client" variabele globaal gemaakt worden. Dit zal nodig zijn om berichten te verzenden buiten de functie die gebruikt wordt om MQTT te initialiseren. De "client" is één van de benodigdheden om de publish-functie te kunnen uitvoeren.

5.4.1 MQTT broker

Gratis MQTT brokers zijn meestal op een of andere manier gelimiteerd. Voor deze toepassing waar meer berichten verstuurd worden per seconde, is dit niet voldoende. Hiervoor zal dus een eigen broker opgezet worden. Hiervoor is een pi gebruikt met een standaard Raspbian OS hierna zijn wat installaties nodig om de MQTT server op te zetten.

- `sudo apt-get update`
- `sudo apt-get upgrade -y`
- `sudo apt-get install mosquitto mosquitto-clients -y`
- `sudo systemctl start mosquito`
- `sudo systemctl enable mosquito`

Nadat deze correct geïnstalleerd zijn, is het connecteren hetzelfde als bij de online brokers. Een voorbeeld hiervan is `"mqtt://192.168.173.205:1883"`. Dit voorbeeld is van een broker die enkel lokaal gehost wordt.

5.4.2 MQTT explorer

Om het overzicht duidelijk te maken en om de MQTT berichten te monitoren wordt er gebruik gemaakt van "MQTT explorer". De berichten worden verdeeld. Het begint met de topic in te stellen als ProjectIOT. Vervolgens wordt er een onderscheid gemaakt tussen de metingen van de modbusberichten en de request die gemaakt worden via de website. Dit is enkel om de zichtbaarheid van de berichten duidelijker te maken, aangezien het modbusbericht elke halve seconde verstuurd wordt.

Er werd gevraagd naar de mogelijkheid om data te versturen via de website. Er is een knop toegevoegd die data verstuurd. Deze knop maakt een post request die de data naar een variable in de C-code brengt. Voorlopig brengt de request geen data met zich mee. De request zelf wordt gedetecteerd en deze detectie zorgt voor het versturen van een hardcoded message.

5.4.3 MQTT berichten versturen

Berichten versturen via MQTT kan gedaan worden met:

```
esp_mqtt_client_publish(client, "ProjectIOT", "connected", 0, 1, 0);
```

De client werd afgeleid uit de configuratiefile. Deze variabele globaal maken is belangrijk om deze buiten de "mqtt_app_start" functie te krijgen en in andere functies een bericht te kunnen versturen.

De tweede waarde is de topic. Dit kan gezien worden als het adres waar het bericht ontvangen wordt.

De derde waarde is het bericht dat verstuurd wordt. In deze functie zal een "const char" doorgegeven worden. Er moet wel opgelet worden met het formaat van de data.

De 4^e waarde is de lengte van het bericht. Als er een 0 doorgegeven wordt, zal het berekend worden aan de hand van de lengte van de data-string.

De 5^e waarde is de quality of service (QoS)

Er zijn 3 niveaus van QoS:

0 - at most once

1 - at least once

2 - exactly once

En tot slot is de 6^e waarde de retain flag. In de normale omstandigheden als niemand "subscribed" is aan de broker, op de juiste topic, wordt het bericht verwijderd. Maar als de "retain flag" op "1" of "True" staat, wordt het laatste bericht bijgehouden.

5.4.4 MQTT berichten ontvangen

Berichten ontvangen via MQTT kan gedaan worden met:

```
esp_mqtt_client_subscribe(client, "ProjectIOT", 0);
```

Ook hier werd de "client" afgeleid uit de configuratiefile.

De tweede waarde is ook hier de topic. Deze waarde kan gezien worden als het adres waar het bericht ontvangen wordt.

De derde en laatste factor die hier ingevuld wordt, is de QoS (quality of service).

Er zijn 3 niveaus van QoS:

0 - at most once

1 - at least once

2 - exactly once

5.5 mDNS

mDNS is een service die ervoor zorgt dat het IP-adres van het apparaat ook “vertaald” kan worden. Een voorbeeld hiervan is aangegeven in het project. In de configuratie file werd nu “Bepowered” doorgegeven. Hierdoor zal het IP-adres veranderen van bijvoorbeeld “192.168.0.172” naar “Bepowered.local”

mDNS maakt het makkelijker om de website die op de ESP32 gehost wordt te vinden in de browser. Het gebruik van Bepowered.local verwijdt de nood aan het zoeken van het IP-adres om de website weer te geven. Zonder mDNS moet er gezocht worden naar het IP-adres.

5.6 Start afgewerkt product

In de configuratie moeten enkele variabelen ingesteld worden voordat het programma correct kan beginnen te werken.

Om te beginnen zijn de WiFi SSID en het WiFi-paswoord nodig om de verbinding met het netwerk te maken. Deze zullen dan ook afhankelijk zijn van de locatie waar de ESP32 op geïnstalleerd wordt.

De volgende waarden zijn de modbusparameters. Ze zijn standaard correct ingesteld bij het initieel laden van de configuratiedata maar toch is het aangeraden om deze te controleren.

- UART port number 1
- UART communication speed 9600
- UART RXD pin number 9
- UART TXD pin number 10
- UART RTS pin number 0
- Modbus communication mode RTU mode

De MQTT broker wordt vervolgens doorgegeven. **mqtt://mqtt.eclipse.org** is de standaard en een gratis broker die momenteel niet meer bereikbaar is. Het is beter om een eigen broker op te zetten. Hiervoor moet dan natuurlijk wel een ander IP-adres gebruikt worden. Beginnend met “mqtt://” gevolgd door het IP-adres bijvoorbeeld “192.168.173.205” met tot slot de juiste port toegevoegd bijvoorbeeld “:1883”.

Als dit voorbeeld gebruikt wordt, moet “mqtt://192.168.173.205:1883” meegegeven worden in de broker URL.

Tot slot wordt de mDNS hostname meegegeven. Dit is standaard ingesteld als “Bepowered”. Door mDNS wordt de “.local” automatisch toegevoegd.

6 Besluit

Doordat de corona-maatregelen het samenkomen beperkten, had het stagebedrijf besloten te telewerken. Dit was geen ideale situatie. Alle communicatie verliep via Skype. Door het telewerken is het gevoel van een werkomgeving totaal anders. Doordat deze stage enkele hardware toepassingen nodig had, heeft er een eenmalig een bezoek plaatsgevonden aan het bedrijf.

Ik had al wel van modbus gehoord maar er zeer beperkt gebruik van gemaakt. Door deze stage heb ik me hier in verdiept. Deze stage heeft me geleerd hoe deze modbusberichten opgebouwd worden.

Websites maken is niet moeilijk maar het inlezen van data die ook live gemeten wordt, is een ander verhaal. Door deze waarde zo dicht bij de realiteit te krijgen, is er onderzoek gedaan. Initieel moest er gezocht worden naar hoe het mogelijk is om met javascript, waarde van een API te halen.

Vervolgens is MQTT toegevoegd. Hier had ik eerder al mee gewerkt maar de grote meerderheid van de projecten waarin dit gebruikt werd, is gecodeerd in Python. Dit wil dus zeggen dat er onderzocht moest worden naar hoe dit werkt in C.

Tot slot is mDNS toegevoegd aan het project. Dit had ik nog niet eerder gebruikt. Dit was een heel nieuw maar handig onderdeel van het project.

De ene dag ging het al wat vlotter dan de andere. Als ik vastliep, probeerde ik eerst op eigen houtje een oplossing te vinden. Daarna nam ik contact op met het stagebedrijf en hielpen zij me verder. Het was een leerrijke stage.

7 Bronnen

7.1 Modbus info

https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

7.2 ESP32 Libraries

7.2.1 Modbus

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/modbus.html>

7.2.2 HTTP server

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_server.html

7.2.3 MQTT

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html>

7.2.4 mDNS

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mdns.html>