In dit readme bestand zou ik graag mijn code wat verder uitlichten. Dit zou ik willen doen door bovenaan de code te beginnen, en verhalend naar het einde van de code te lopen. Hierna zou ik elke class willen uitlichten om zo te benadrukken hoe deze werken.

Omdat ik de beschrijving van de code zo aanvlieg, is het slim om de code naast dit bestand te houden. Ik zal in mijn verhaal de regelnummers gebruiken. Echter is het toch handig om naast dit verhaal de code visueel op uw scherm te hebben.

*Main programma*

In de eerste regels roep ik de benodigde libraries aan. Deze zijn nodig voor de verschillende functionaliteiten waar mijn code hevig gebruik van maakt. Op regel 12 beginnen de includes voor mijn header files. Dit doe ik zodat ik mijn classes in aparte bestanden kan maken. Zo blijft de main code file zo netjes mogelijk.

Hierna op regel 21 een define. Ik ben afgeraden defines te gebruiken, echter werkt de library voor de dht sensor op een define.

Op regel 24 beginnen de const int aanroepingen die ik nodig heb om mijn classes te initialiseren. Door hier variabeles van de maken is het makkelijk hier later aanpassingen in te doen.

Op regel 39 begint het initialiseren van mijn classes. In de parameters gebruik ik de hiervoor gedeclareerde variabeles om de pins van sensoren te declareren. De lichtsensor class heeft nog wat extras, de naam van de sensor die later wordt geprint. Zo kan ik de functie binnen de class voor het uitprinten van waardes simpel houden.

De volgende regels bevatten verschillende wifinetwerken. Afhankelijk van de locatie kan hier worden gecomment of ge-uncomment om zo het juiste netwerk te selecteren.

Op regel 59 zien we dan een url. Dit is het eindadres waar de ESP32 de HTTP post naar toe stuurt. Zoals u kunt zien is dit het domein van een groepsgenoot, zijn naam staat er in.

Op regel 62 vinden we de eerste functie. Deze functie bevat een class. Het einddoel van de functie is om ten eerste te bepalen: welk percentage grondvochtigheid is oke? (Oftewel, wanneer hoef ik geen water te geven?) En wanneer ik water moet geven, dan voer ik pas dit stukje code uit. Zo is te zien in de if statement.

Op regel 79 begint de functie voor het bedienen van de indicator led. Deze led geeft aan hoe vol het reservoir is. Ook deze functie maakt gebruik van een class.

Op regel 96 hebben we een functie gemaakt om de loop op te delen in kleinere stukjes. Deze functie is verantwoordelijk voor het groeperen van alle class aanroepingen die in de loop data van de sensoren opvragen.

Op regel 110 hebben we een functie gemaakt om de loop op te delen in kleinere stukjes. Deze functie is verantwoordelijk voor het groeperen van alle class aanroepingen die in de loop de data van de sensoren verwerkt.

Op regel 122 hebben we een functie gemaakt om de loop op te delen in kleinere stukjes. Deze functie is verantwoordelijk voor het groeperen van alle class aanroepingen die in de loop de data van de sensoren uitprint.

Op regel 137 hebben we een functie gemaakt om de loop op te delen in kleinere stukjes. Deze functie is verantwoordelijk voor het groeperen van alle class aanroepingen die in de loop de verwerkte data uitprint.

Na deze functies kan op regel 150 de void setup beginnen. Deze setup bevat een aantal normale dingen zoals het bepalen van de baud rate en een tweetal libraries initialiseren. Ook de verbinding met het wifi netwerk wordt in de setup gedaan.

Dan op regel 166 de void loop, dit is waar het allemaal gebeurt. In het begin worden er wat tijdsindicaties opgegeven. Dit heeft te maken met de timing van de post in de loop. Ook worden er twee functies aangeroepen, deze hebben niks te maken met de post dus heb ik deze buiten die if statement gelaten. Een ander belangrijk ding is het volgende. Als deze in de if zouden staan zou de plant geen water krijgen als de wifi uitvalt. Door deze oplossing krijgt de plant altijd water, ook al zou de wifi uitvallen.

Dan de if statements waar ik het over had. De eerste checkt de tijd en de intervals, de tweede zorgt ervoor dat een post alleen gemaakt kan worden als er wifi is. Wel zo handig.

Duiken we dieper in deze loop dan komen we aan bij een aantal functies. Door de gekke manier hoe de dht library is opgemaakt is het nodig dat ik regel 79 en 180 buiten een functie neer moet zetten. Anders zou ik twee aparte functies aan moeten maken met elk een return value, dat is natuurlijk een onnodige omweg. Daarna een viertal functies die elk een basisfunctie groeperen. Het opvragen van data, het verwerken van de data, het printen van de rauwe data en het printen van de verwerkte data.

Op regel 187 volgt dan het omzetten van alle data naar een string. Dit wordt gedaan met behulp van classes. Dit moet gebeuren vanwege de JSON string die alleen met het datatype string om kan gaan.

Op regel 196 volgt de setup van de HTTP post. Een aantal declaraties op de eerste 5 regels. De headers zijn hier van belang. In deze headers geven we namelijk de authenticatie gegevens mee voor de webserver. Deze kan onze HTTP post zo onderscheiden van derden. Hierna volgt de JSON string die verzonden wordt. Veel gekke tekens, de webserver filtert deze er echter uit.

De volgende regels (202 tot 205) handelen het antwoord van de website en de webserver af om ze zo netjes weer te kunnen geven, dit is eigenlijk alleen handig voor debugging en troubleshooting. Echter is het dus wel goed erin te houden.

Op regel 208 begint het leegmaken van alle strings om deze klaar te maken voor de volgende loop. Dit is nodig omdat de nieuwe waardes anders gewoon achter de vorige waardes worden gezet. Alle data gaan dan dus verloren.

Dan als laatst hebben we een failsafe voor als de wifi disconnect. De laatste stukjes code zorgen ervoor dat dit wordt weergegeven en dat het programma weer probeert verbinding te zoeken met het wifi netwerk.

*Classes*

In dit stukje zou ik graag alle .cpp files uitlichten. De .h files zijn namelijk heel erg vanzelfsprekend bij het opbouwen van classes. Hier is dus niks speciaals over te vertellen. Ik zou dus liever vertellen hoe ik de rekenkundige problemen heb opgelost. Ik zal refereren naar functie, hiermee bedoel ik natuurlijk de functies binnen een class.

GrondVochtigheid.cpp

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pin mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken.

De tweede functie op regel 10 zorgt ervoor dat er via een analogread data wordt opgevraagd van de sensor.

De derde functie op regel 17 zorgt ervoor dat er via een berekening een procentuele waarde uit de opgevraagde data komt.

De vierde functie op regel 24 zorgt ervoor dat de rauwe data wordt geprint in de console.

De vijfde functie op regel 30 zorgt ervoor dat de verwerkte data wordt geprint in de console.

De zesde functie op regel 36 zorgt ervoor dat de verwerkte data in een string wordt gezet.

De zevende functie op regel 43 zorgt ervoor dat de string leeg wordt gemaakt.

*LedController.cpp*

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pins mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken. In dit geval gebruik ik twee pins, dit komt omdat een RGB led zo werkt.

De tweede functie op regel 13 zorgt ervoor dat de led groen zal branden.

De derde functie op regel 19 zorgt ervoor dat de led rood zal branden.

*LichtSterke.cpp*

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pin mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken. Ook geef ik hier de naam van de sensor mee. Dit doe ik omdat ik op deze manier later gemakkelijk mee kan printen welke sensor welke data verstrekt.

De tweede functie op regel 11 verzamelt de data via een analogread.

De derde functie op regel 17 zorgt ervoor dat er via een berekening een procentuele waarde uit de rauwe data komt.

De vierde functie op regel 24 zorgt ervoor dat de rauwe data in de console geprint wordt.

De vijfde functie op regel 32 zorgt ervoor dat de verwerkte data in de console geprint wordt.

De zesde functie op regel 40 zet de verwerkte data in een string.

De zevende functie op regel 47 maakt deze zelfde string weer leeg.

*MyDHT.cpp*

Aan de naam van deze class valt misschien iets op, er staat my in. Dit duid op het feit dat deze class gebruik maakt van inheritance. De originele class is dht.cpp, door inheritance neemt mijn class alles van deze originele class over PLUS het geen wat ik nog toevoeg. In dit geval zijn dit de volgende functies

De eerste functie op regel 4 wordt gebruik om de class aan te roepen, in de parameters staan de pin, het type dht sensor en de count. Deze count staat standaard op 6.

De tweede functie op regel 9 zet de rauwe data verzameld door de class om naar een string. Dit was een lastige class om te maken aangezien de dht sensor twee waardes teruggeeft.

De derde functie op regel 18 maakt de string weer leeg.

De vierde functie op regel 25 print de temperatuur in de console.

De vijfde functie op regel 32 print de luchtvochtigheid in de console.

*PotentioMeter.cpp*

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pin mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken.

De tweede functie op regel 10 vraagt de data op doormiddel van een analogread().

De derde functie op regel 17 zet de rauwe data om in een procentuele waarde.

*UltrasonicSensor.cpp*

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pin mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken.

De tweede functie op regel 12 vraagt de rauwe data van de sensor.

De derde functie op regel 25 zet deze rauwe data om in een procentuele waarde. Deze berekening was erg ingewikkeld aangezien ik iets moest verzinnen waarbij de diepte van de pot 100% zou zijn. Het duurde dus even voordat ik deze berekening had verzonnen.

De vierde functie op regel 32 print de rauwe data in de console

De vijfde functie op regel 38 print de verwerkte data in de console.

De zesde functie op regel 44 zet de verwerkte data in een string.

De zevende functie op regel 51 maakt deze string weer leeg.

*WaterPump.cpp*

De eerste functie op regel 4 wordt gebruikt om de class aan te roepen. Hier geef ik in de parameter de pin mee. De this -> betekend dat ik de class meerdere keren aan zou kunnen roepen maar het pin nummer alleen binnen 1 aanroeping blijft. Zo kan je meerdere classes maken zonder dat ze in de war raken.

De tweede functie op regel 10 zet de pomp heel even aan zodat hij gaat pompen.