

TECHNISCH INSTITUUT SINT-PAULUS
Kruisven 25 - 2400 Mol

Geïntegreerde proef

Schooljaar 2017-2018

Studierichting: Elektriciteit-Elektronica

Home Automation (IoT)

Lian Aarts, 6EE

TECHNISCH INSTITUUT SINT-PAULUS
Kruisven 25 - 2400 Mol

Geïntegreerde proef

Schooljaar 2017-2018

Studierichting: Elektriciteit-Elektronica

Home Automation (IoT)

Lian Aarts, 6EE

Woord vooraf

Tijdens de voorbereiding van mijn geïntegreerde proef (GIP) heb ik veel ervaring opgedaan. Mijn mentors hadden mij een onderwerp voorgesteld waar ik zelf weinig voorkennis van had. Na een beetje opzoekwerk geraakte ik gefascineerd door dit potentiële onderwerp en besloot er zonder aarzelen aan te beginnen. Aangezien dit iets was dat mij interesseerde ben ik er in mijn vrije tijd veel mee bezig geweest.

Alvast bedankt aan de heer K. Verholen en de heer G. Daniëls die een origineel onderwerp gekozen hebben en mij tijdens de lessen geholpen hebben om mijn GIP te realiseren. Ook een dank woordje voor mijn ouders die mij zowel moreel als financieel ondersteund hebben.

Inhoudsopgave

1 Inleiding	3
2 Home automation	5
2.1 <i>Raspberry Pi</i>	6
2.2 <i>Actuatoren en sensoren</i>	7
2.3 <i>Software</i>	7
3 Progressie	8
3.1 <i>RGB-webserver</i>	9
3.2 <i>Z-wave</i>	9
3.3 <i>DIY-sensoren en actuatoren</i>	12
3.3.1 <i>RGB-controller</i>	13
3.3.2 <i>Switches</i>	15
3.3.3 <i>Multisensor</i>	17
3.3.4 <i>Dimmer</i>	19
3.4 <i>Andere Widgets</i>	23
3.4.1 <i>Device tracker</i>	24
3.4.2 <i>Nest</i>	25
3.5 <i>Apple Homekit</i>	26
3.6 <i>Automatie</i>	29
3.7 <i>Ondervonden problemen</i>	31
4 Besluit	33
5 Bibliografie	34
6 Bijlage	36
6.1 <i>configuration.yaml</i>	36
6.2 <i>Arduino code dimmer</i>	54
6.3 <i>Arduino code rgb controller</i>	59
6.3 <i>Arduino code rgb controller</i>	73

1 Inleiding

Het project omvat een variatie aan activiteiten die gebruik maken van beweging, licht, geluid en tactiele feedback. Deze activiteiten moet ik introduceren in een variëteit van basisprincipes in de branche van programmering, elektriciteit, elektronica, wiskunde en fysica. Dit project zal onderdelen bevatten van praktische ontwerpen, gebruikt door ingenieurs en techniekers in het creëren van moderne machines en applicaties met behulp van standaard prijsbewuste onderdelen. Steeds meer nieuwe apparaten en technologieën komen elke dag bij ons thuis terecht. Hoewel ze allemaal gericht zijn op het verbeteren van onze levensstijl, missen ze allemaal één belangrijk kenmerk: een gemeenschappelijke taal waardoor ze met elkaar kunnen communiceren om een echte geautomatiseerde en slimme omgeving te creëren. Het hoofddoel van Home Assistant is om een integratieplatform aan te bieden om dit probleem op te lossen.

Als er één ding is dat ons leven makkelijker heeft gemaakt is dit zeker de implementatie van elektriciteit in ons dagelijks leven. Vanaf het einde de 19^{de} eeuw is elektriciteit essentieel in de moderne samenleving. Niet alleen in je eigen thuisnetwerk, maar ook in ziekenhuizen, rustenhuisen,... In deze ziekenhuizen en rusthuizen worden al langer domoticsystemen gebruikt om het leven van de patiënten makkelijker te maken. Sinds enkele jaren geleden wordt dit ook toegepast in woningen om enkelen handelingen makkelijker te maken. Domotica is een systeem wat een enorm prijskaartje met zich meedraagt, ook de bedrading van een domoticsysteem kan men enkel plaatsen in nieuwbouw. Nu is er een nieuw concept op de markt. Via open source software en een microcontroller kan je via een Wi-Fi-netwerk draadloos sensors en actuatoren met elkaar verbinden. Door deze nieuwe software hoeft de klant zich geen zorgen meer te maken in verband met het uitslijpen van muren. Je bent ook niet gebonden aan één enkel systeem, je kan

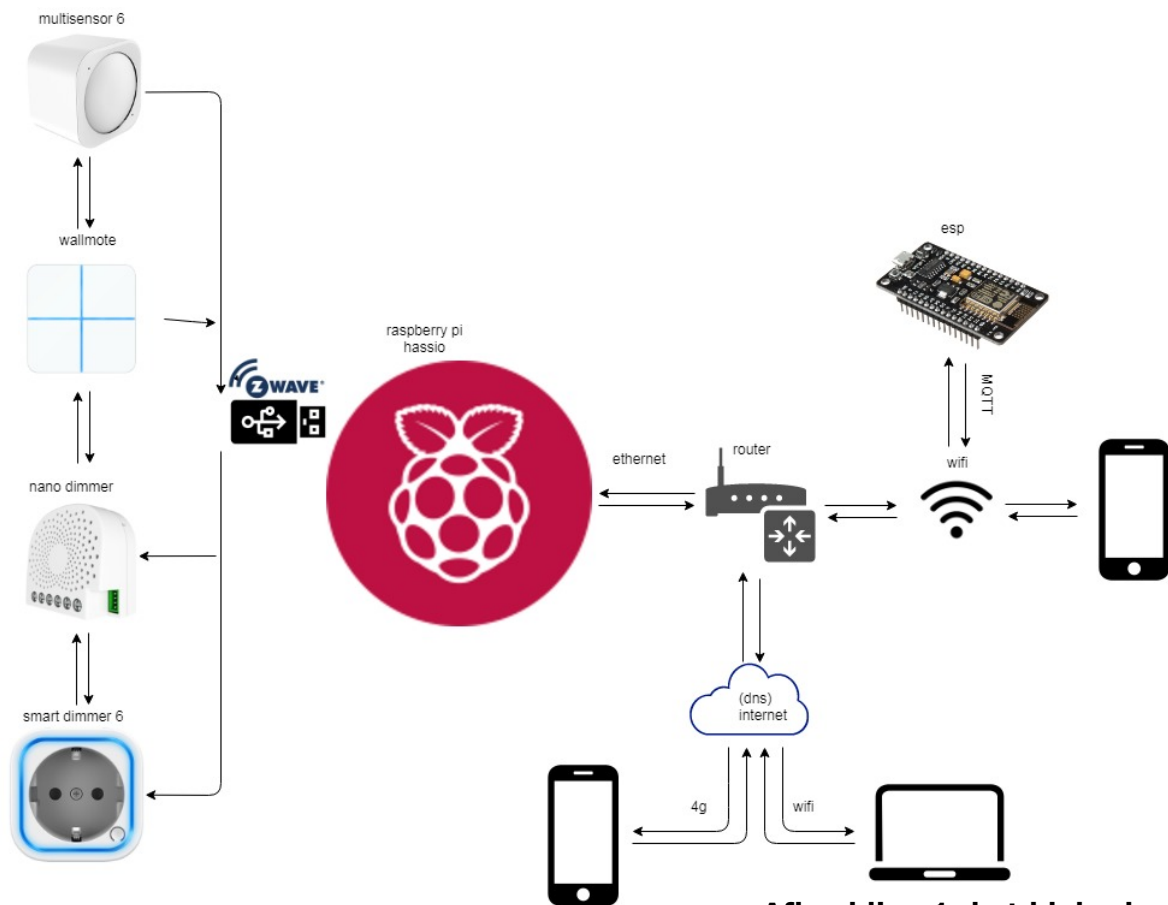
apparaten van verschillende merken verbinden. Dit concept ga ik bestuderen en uitwerken.

Een thuis-netwerk automatiseren op basis van open-source software tegen een betaalbare prijs zonder extreme aanpassing van het bestaande netwerk. Dit systeem moet het levensgemak aanzienlijk verbeteren. De software moet ook makkelijk te hanteren zijn zodat het door jong en oud gebruikt kan worden.

Om de werking van mijn GIP te kunnen waarderen is een voorkennis van programmatie, communicatie en elektronica nodig. Ook een voorkennis van elektriciteit en netwerkbeheer in een huis installatie is aangeraden.

In hoofdstuk 1 introduceer ik het project als geheel. In dit deel zal ik uitleg geven over de samenhang van mijn project. Ik zal ook enkele motieven geven, die mij tot dit project aangesproken hebben. Vervolgens worden de verschillende componenten besproken die deel uit maken van dit systeem. De werkwijze zal ook aangehaald worden.

2 Home automation



Afbeelding 1: het blokschema

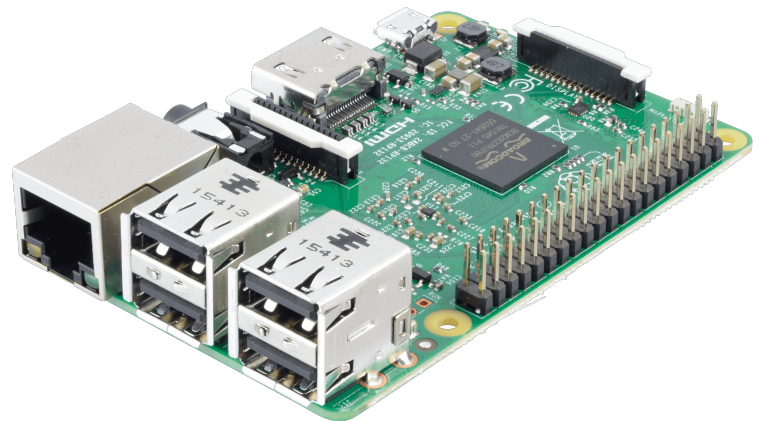
Zoals vermeld in de inleiding zal ik het project in zijn geheel introduceren. In afbeelding 1 kan je duidelijk zien dat de Raspberry Pi de centrale component van de opstelling is. Alles zal via de Raspberry bestuurd en gelezen worden. Alle sensoren en actuatoren zijn op specifieke manieren verbonden met de Raspberry Pi. Dit kan gaan van radiofrequentie tot een verbinding via het globale internet met behulp van een DNS. Wanneer het op de actuatoren en sensoren aankomt, heb ik verschillende opties gebruikt: commerciële en zelfgemaakte apparaten. Ik hou rekening met de kostprijs en met de gebruiksvriendelijkheid die nodig is om alles te configureren en aan te sturen. Bij commerciële producten is het configureren heel simpel en vanzelfsprekend, er zijn duidelijke handleidingen aanwezig op de website van Home Assistant. Als je echter eenmaal aan het "DIY" component begint, wordt het wat ingewikkelder. Op de website zijn geen duidelijke tutorials te vinden. Je zal zelf moeten zoeken naar oplossingen. Vele problemen kan je oplossen door het Home Assistant forum te raadplegen.

Op Youtube kan men ook verschillende tutorials terugvinden, maar éénmaal alles verbonden is, zal de gebruiksvriendelijkheidgraad dezelfde zijn. Het grote voordeel van de zelfgemaakte apparaten is de kostprijs, want je kunt vaak voor één tiende van de prijs hetzelfde effect verkrijgen. Wil je er veel tijd in steken en heb je ervaring met elektronica? Ga dan zeker voor de DIY-optie.

Je hebt dus de 3 grote delen: de Raspberry Pi, de actuatoren en de sensoren en het besturingssysteem Home Assistant. Deze componenten zal ik in meer detail bespreken in volgende hoofdstukken.

2.1 Raspberry Pi

De Raspberry Pi is een “single board computer”, dit wil zeggen dat de computer gebouwd is op één enkele printplaat. De microprocessor, het geheugen en de i/o zijn aanwezig op de printplaat.



Afbeelding 2: Raspberry Pi :

Voor je de Raspberry Pi opstart, zal je een besturingssysteem moeten installeren. Er wordt meestal via een softwareprogramma een image geflasht op een micro-SD-kaartje. In mijn geval is dit de software van Home Assistant. Wanneer de software geflasht was, heb ik de files op de micro-SD-kaart aangepast. Ik heb de bestanden aangepast zodat de Raspberry Pi een statisch IP-adres heeft. Hierdoor kan je makkelijk een wifi-verbinding maken met de actuatoren en sensoren.

Er zijn ook enkele GPIO-pinnen (general-purpose in/out-pinnen) beschikbaar op de Raspberry Pi. Deze kan je ook aansturen via de software van Home Assistant. Dit is natuurlijk niet handig omdat het niet draadloos is. Verder zijn er ook standaard in/out-poorten beschikbaar zoals usb, ethernet, ...

2.2 Actuatoren en sensoren

De actuatoren en sensoren vormen belangrijke componenten van het eindwerk. Zonder deze componenten kan je natuurlijk je huis niet automatiseren. Er zijn verschillende opties beschikbaar. Home Assistant ondersteunt 975 verschillende apparaten, dit gaat van simpele schakelaars tot voice control. In elke update die uitgebracht wordt, implementeert men tientallen nieuwe opties. Je kan zelfs je Tesla verbinden met Home Assistant en zo de locatie bekijken en zelfs de deuren sluiten. Elke component heeft op de website van Home Assistant een eigen handleiding. Bij sommige zijn de nodige handelingen natuurlijk makkelijker dan bij andere.

Wanneer je zelf sensoren gaat maken, zijn er natuurlijk ook verschillende opties. MQTT is één van de bekendste opties. Door MQTT kan je dus sensoren en actuatoren maken naargelang de behoefte van de klant. Dit is dan ook het grote voordeel van zelfgemaakte componenten, want je kan zelf alles aanpassen en maken wat je wilt.

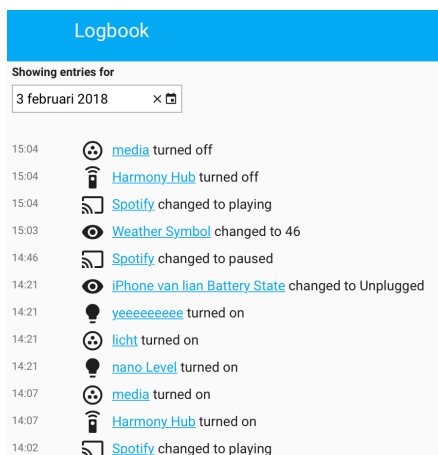
2.3 Software

Home Assistant is de populairste open source "home automation" optie. Ik heb Home Assistant gekozen omdat er zo veel ondersteunde componenten zijn. Home Assistant heeft ook een applicatie die je gratis kan downloaden en verbinden met de Raspberry Pi. Alle verschillende applicaties die ik nodig had, heb ik nu door Home Assistant vervangen. Zelf de televisie is verbonden met Home Assistant.



Afbeelding 3: Home Assistant app

Je hebt volledige controle over alle componenten, alles wordt gevisualiseerd in grafieken en alle veranderingen worden weergegeven in het logboek. Home Assistant implementeert alle componenten heel mooi in de lay-out. Je kan zelf componenten onderverdelen in groepen en de lay-out helemaal aanpassen. De software hoeft ook niet noodzakelijk op een Raspberry Pi geïnstalleerd te worden, je kan ook een oude computer gebruiken. Ik heb voor de Raspberry Pi gekozen omdat dit de goedkoopste optie is. Er zijn verschillende "add-ons" beschikbaar. Dit zijn geen echte componenten maar ze vergroten de



Afbeelding 4: het logboek

functionaliteit van de software. Samba Share is bijvoorbeeld een add-on die je kan gebruiken om de configuratiebestanden aan te passen via je thuisnetwerk. Er is ook een add-on om een lokale MQTT broker te maken op je Raspberry. Het logboek (afbeelding 4) is ook een belangrijk deel van Home Assistant. Hierin kan je alle statusveranderingen bekijken. In het debuggen is dit heel handig om fouten te zoeken.

3 Progressie

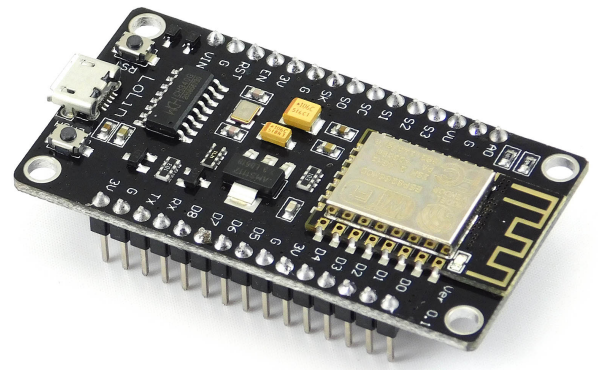
Met veel enthousiasme ben ik aan het project begonnen. Maar omdat er nog geen materiaal besteld was, kon ik niet meteen aan de slag. Mijn vader had thuis nog een Raspberry Pi 1 liggen die hij niet gebruikte. Op deze Raspberry Pi, heb ik dan de nodige software geïnstalleerd. Na verschillende testen ben ik tot het besluit gekomen dat het communicatieprotocol MQTT niet werkt op deze oudere Raspberry Pi. Wanneer ik zelf een Raspberry Pi 3 aangekocht had, viel het me snel op dat deze vele malen krachtiger was. Het installeren en opstarten verliep vele malen sneller. De MQTT Broker startte ook zonder problemen op.

3.1 RGB-webserver

Ik had nog altijd geen sensoren en actuatoren dus begon ik te experimenteren met een esp. De esp8266 is een wifi-module die je kan verbinden met je router. Door deze chip kan je microcontrollers verbinden met een netwerk.

Er zijn ook varianten beschikbaar waarin al een esp-chip (in een microcontroller) verwerkt zit (afbeelding 5). Ik heb deze variant gebruikt omdat het voordeliger en makkelijker te gebruiken is.

Ik heb de esp eerst geflasht met de software van nodeMCU, zodat ik de module kon verbinden met de computer. Daarna heb ik met het programma ESPlorer een code geüpload die te vinden is op het internet (<http://randomnerdtutorials.com>). Je moet de code wel aanpassen door je ssid en je wachtwoord bij in de code te typen zodat je de esp met je lokaal netwerk kunt verbinden. Eenmaal de code gestart is, zal er een IP-adres verschijnen in de seriële monitor. Wanneer je het IP-adres in je favoriete browser typt, zal er een webpagina verschijnen waarmee je de kleur van de led kan veranderen.



Afbeelding 5: Node MCU

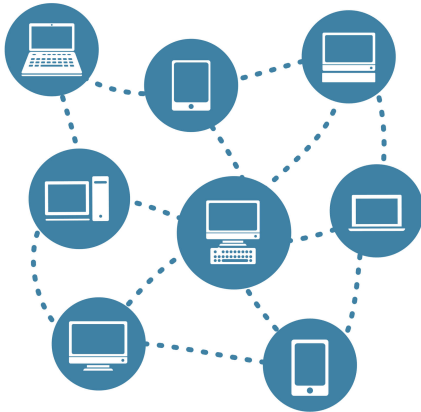
3.2 Z-wave

Z-wave is een communicatieprotocol dat voornamelijk gebruikt wordt in home automation. Er wordt door verschillende apparaten een "mesh network" gevormd van radiogolven met een laag vermogen. Het communicatieprotocol wordt in allerlei toepassingen gebruikt zoals: licht, beveiligingssystemen, jaloezieën, ... Zoals in andere protocollen wordt alles bestuurd



**Afbeelding 6:
Aeotec Z-stick**

via een centrale controller. In mijn geval is dat de Aeotec Z-stick (afbeelding 6). Dit is een kleine usb-stick waarmee je alle Z-wave-apparaten verbindt. Deze usb-stick wordt dan verbonden met de Raspberry Pi.



Afbeelding 7: voorbeeld mesh network

Omdat de Z-wave-apparaten zelf radiogolven uitstralen, is er geen nood aan range extenders. Wanneer de controller het apparaat niet kan bereiken, zal een ander apparaat het signaal versterken. Zo wordt er een "mesh network" (afbeelding 7) gevormd waar apparaten met elkaar communiceren om op deze manier informatie met elkaar te delen. Het enige nadeel van Z-wave is de kostprijs. Z-wave is zeker niet de voordeligste optie, maar het configureren is vanzelfsprekend, de producten zijn altijd mooi afgewerkt en je bent niet merkgebonden. Er zijn tientallen producenten die het Z-wave-protocol gebruiken, maar in tegenstelling tot andere home-automation-systemen werken alle apparaten vlekkeloos samen.

Zoals eerder vermeld heb ik een Aeotec Z-stick aangekocht. Natuurlijk moet je ook een apparaat hebben om aan te sturen, dus heb ik ook de Aeotec nano-dimmer aangekocht. De dimmer is ontworpen om in een bestaande contactdoos te passen en gebruik te maken van een gewone wandschakelaar. Ik heb eerst een

testopstelling (afbeelding 8) gemaakt zodat ik de automaat niet constant moest uitschakelen. Bij het aansluiten heb je 2 keuzes: je kan de schakelaar aansluiten met nulgeleider of zonder nulgeleider.

Omdat er maar 1 draad onderbroken werd door de schakelaar en de andere niet

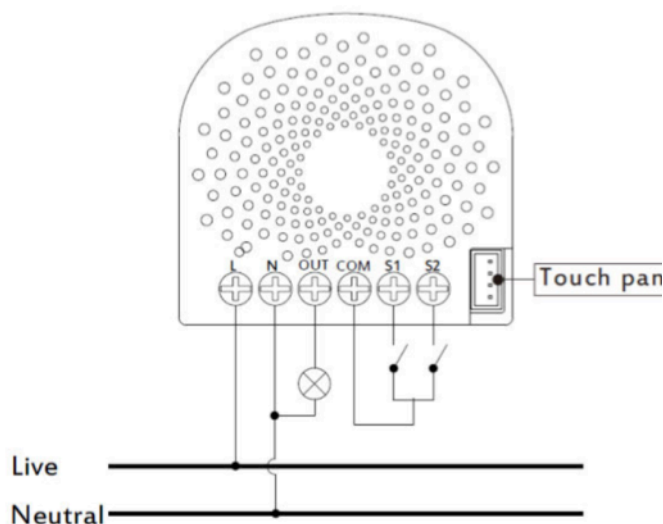


Afbeelding 8: testopstelling Nano-Dimmer

beschikbaar was, heb ik voor de schakeling zonder nul geleider gekozen. Wanneer ik de dimmer dan éénmaal in de contactdoos geïnstalleerd had, werkte de schakeling niet. Omdat ik een laag vermogen Led filament lamp gebruikte, heb ik een gloeilamp getest. De schakeling werkte. Na enkele minuten denkwerk besloot ik om opnieuw een testschakeling te maken, maar nu wel met de nulgeleider aangesloten.

Deze schakeling werkte wel met de ledverlichting.

Het probleem was dat ik geen nulgeleider beschikbaar had in de contactdoos. Ik heb stroom van een onderliggend stopcontact afgetakt en zo heb ik dan het aansluitschema met nulgeleider gekozen (afbeelding 9). Dit is geen veilige oplossing, omdat de automaten van de stopcontacten en van de lichten nu beide uitgeschakeld moeten worden om veilig aan de lichtschakelaar te werken. Nadat de module geïnstalleerd was, moest ik de schakelaar nog aansluiten, hiervoor is een verbindingsproces nodig zodat de module weet welke soort schakelaar je gebruikt.



Afbeelding 9: schema met nul geleid

```
zwave:  
  usb_path: /dev/ttyACM0
```

**Afbeelding 10: deel uit de
configuration.yaml file**

Wanneer je een nieuw apparaat wilt integreren in Home Assistant, moet je de "configuration.yaml" file aanpassen naargelang het apparaat. De aanpassing die nodig is voor Z-wave is beperkt. Er moeten enkel 2 kleine regeltjes toegevoegd worden (afbeelding 10). De eerste regel implementeert de Z-wave-functionaliteit in Home Assistant. In de tweede regel vertel je waar de Z-stick aangesloten is. Eénmaal de regels toegevoegd zijn en de Raspberry herstart is, zal het device tevoorschijnkomen op de voorpagina. Als alles goed aangesloten is, zou de lamp nu bestuurbaar zijn via Home Assistant.

3.3 DIY-sensoren en actuatoren

Een ander aspect van de opdracht is dat ik zelf sensoren en actuatoren zou maken. Wanneer er over DIY (do it yourself) wordt gesproken in verband met Home Assistant, zal dit meestal gaan over een esp die verbonden is via een wifi-verbinding. Wanneer je zelf een sensor of actuator maakt, zal de kostprijs niet groot zijn. Je kan vaak voor een aanzienlijk kleinere prijs een apparaat met dezelfde functionaliteit maken. Ik maak gebruik van een lokale MQTT broker die door een add-on geïntegreerd kan worden in de software van Home Assistant.

MQTT is een “machine to machine” communicatieprotocol. Het maakt gebruik van een publish-subscribe architectuur. Dit wil zeggen dat er één centrale bestuurder nodig is, een zogenaamde “broker”. Het grote voordeel van MQTT is dat de communicatie het netwerk niet hard belast, het communiceren is dus mogelijk in huizen waar geen goed netwerk aanwezig is. Er is een kleine configuratie (afbeelding 12) nodig voor de Mosquitto broker. Je kan de broker beveiligen door een wachtwoord en username toe te voegen (weggelaten voor privacyredenen). Hier is het

```
const char* WIFI_SSID = " ";
const char* WIFI_PASSWORD = " ";

// MQTT: ID, server IP, port, username and password
const PROGMEM char* MQTT_CLIENT_ID = "office_rgb_light";
const PROGMEM char* MQTT_SERVER_IP = "192.168.0.50";
const PROGMEM uint16_t MQTT_SERVER_PORT = 1883;
const PROGMEM char* MQTT_USER = " ";
const PROGMEM char* MQTT_PASSWORD = " ";
```

Afbeelding 12: Arduino code

Options

```
{
  "plain": true,
  "ssl": false,
  "anonymous": true,
  "logins": [
    {
      "username": ,
      "password": 
    }
  ],
  "customize": {
    "active": false,
    "folder": "mosquitto"
  },
  "certfile": "fullchain.pem",
  "keyfile": "privkey.pem"
}
```

SAVE

Afbeelding 11: configuratie van de add-on

ook belangrijk om een statisch IP-adres te hebben, want je verbindt je apparaat door

middel van het IP-adres. Als je IP-adres van de broker verandert, zullen de apparaten niet meer verbinden met de Raspberry Pi.

Er zijn ook vaak configuratielijnen die je in de Arduino-code moet aanpassen om het apparaat te verbinden. Je zal de nodige informatie moeten invullen om met het Wi-Fi-netwerk te verbinden en om het apparaat met de MQTT broker te verbinden (afbeelding 11). In mijn geval is dit het IP-adres van de Raspberry Pi. Het andere belangrijke deel van de code zijn de topics. Deze topics moeten beide in de code van het apparaat en in de configuratiefile van de Raspberry Pi vermeld staan voor een goede werking. De topics zorgen ervoor dat de Raspberry weet welke functionaliteit het apparaat bezit. Wanneer een apparaat meerdere variabelen heeft zullen er meer topics nodig zijn. Een simpele schakelaar heeft enkel 2 topics nodig om te kunnen werken. Een RGB controller heeft daarentegen 6 topics nodig.

3.3.1 RGB-controller

Ik heb enkele apparaten gemaakt. Ik ben begonnen met het maken van een RGB-controller die dezelfde gebruiksvriendelijk en functionaliteit heeft als andere aangekochte oplossingen. Hiervoor zijn enkele standaard programma's aanwezig die je kan gebruiken. Zoals eerder verteld hoef je vaak alleen de wifi gegevens te veranderen. In de geteste opstelling gebruikte ik een common kathode RGB-led die ik met voorschakelweerstand verbond met de microcontroller. Deze opstelling werkte zoals gewenst, ik kon via de software van Home Assistant de led besturen. Omdat ik deze opstelling wou gebruiken om een RGB-strip aan te sturen heb ik nog andere componenten moeten gebruiken om de grote stromen te sturen. Ik heb een npn transistor (TIP120) gebruikt om via de lage stuurspanning een 12 volt RGB-strip aan te sturen.

Om deze opstelling gebruiksvriendelijk te maken heb ik een PCB gemaakt die je enkel met een stopcontact moet verbinden om de RGB-strip operationeel te maken. De nodeMCU kan je voeden met een spanning tussen 3,3 volt en 20 volt, daardoor heb ik geen extra omvormer moeten gebruiken. Bij het maken van de print heb ik gebruik gemaakt van een CNC-machine die "gerber" files gebruikt om zo het nodige koper weg te frezen. Eénmaal het koper is weggefreest, zullen er koperbanen overblijven. Bij het solderen heb ik enkel problemen ondervonden. De gefreesde banen zijn omringd door overtollig koper dat niet gebruikt wordt. Het tin gaat natuurlijk ook hier willen plakken, daarom is het belangrijk om alle verbindingen te meten zodat er geen foute verbindingen worden gemaakt. De print was gesoldeerd maar werkte nog niet. De transistors die ik besteld had waren nog niet aangekomen, dus had ik andere transistors gebruikt die ik bij de hand had. Deze had ik spijtig genoeg niet getest en ze werkten uiteindelijk ook niet. Een beetje geduld kan je heel veel werk en moeite besparen, dit heb ik spijtig genoeg uit mijn fouten moeten leren. De transistors werden namelijk dezelfde dag nog geleverd. Na veel soldeerwerk heb ik de transistors kunnen vervangen. Met een zucht van opluchting zag ik de schakeling werkte.

Nadat ik de RGB-strip gebruikt had in een "automation", kwam ik tot de conclusie dat de functie "transition" niet werkt. Deze functie zorgt ervoor dat de lichtintensiteit toeneemt op een bepaalde tijd. En omdat ik de RGB-strip wil gebruiken als oriëntatielicht dat 's nachts door beweging ingeschakeld wordt, is dit wel een leuke functie.

Na een beetje onderzoek kwam ik op de website van Home Assistant een tabel tegen waarin alle bruikbare functies voor de verschillende lichtplatformen weergegeven zijn.

Omdat men in het platform “MQTT Json” de functie “transition” wel kan gebruiken heb ik deze geprobeerd. Maar omdat ik al een PCB had gemaakt moest ik hetzelfde schema gebruiken. De nieuwe code functioneerde goed maar ik kon de functie “transition”

Function	mqtt	mqtt_json	mqtt_template
Brightness	✓	✓	✓
Color temperature	✓	✓	✓
Effects	✓	✓	✓
Flashing	✗	✓	✓
RGB Color	✓	✓	✓
Transitions	✗	✓	✓
XY Color	✓	✓	✗
White Value	✓	✓	✓

Afbeelding 13: tabel met functionaliteiten

nog altijd niet gebruiken. Ik heb de forum geraadpleegd en kwam te weten dat de functie “transition” alleen bruikbaar is bij het veranderen van kleuren.

3.3.2 Switches

Ik heb ook schakelaars gemaakt met MQTT. De werking is gelijkaardig met die van de RGB-strip, dus zal ik hier de code uitgebreider bekijken en uitleggen. Alle MQTT-codes hebben dezelfde opbouw. Er zijn enkele standaard MQTT-programma’s aanwezig op de website van Github. Deze programma’s kan je makkelijk aanpassen.

ha_mqtt_binary_sensor_ble_scanner
ha_mqtt_binary_sensor_door
ha_mqtt_binary_sensor_nfc_scanner
ha_mqtt_binary_sensor_pir
ha_mqtt_light
ha_mqtt_light_with_WiFiManager_mDNS_and_OTA
ha_mqtt_light_with_brightness
ha_mqtt_multisensor
ha_mqtt_rgb_light
ha_mqtt_rgbw_light_with_discovery
ha_mqtt_sensor_dht22
ha_mqtt_sensor_photocell
ha_mqtt_switch

Afbeelding 14: standaard MQTT programma's

Zoals eerder vermeld hebben alle programma's een gelijkaardige opbouw. Er is altijd een deel dat aanwezig is om de nodige verbindingen te maken met de Raspberry Pi, een deel dat zorgt voor de communicatie en een deel dat de hardware gaat aansturen. Voor de wifi-verbinding is de ESP8266WiFi library noodzakelijk. In enkele lijnen code kan je verbinden met je lokale wifi-netwerk. De communicatie is dan weer mogelijk via de topics die al eerder besproken zijn. De hardware wordt aangestuurd wanneer de esp8266 een bericht ontvangt. Als het bericht de juiste topics bevat zal de esp pas een taak uitvoeren. Wanneer er een taak uitgevoerd is, zal de esp ook een bevestiging geven aan de broker dat deze taak wel degelijk is uitgevoerd.

In beide codes zijn dezelfde topics gebruikt, dit is nodig voor de juiste werking. Er zijn twee topics die gebruikt worden. De "/house/switch1" is de topic die de broker zal veranderen en verzenden naar de verschillende apparaten. De esp ontvangt het bericht. Enkel als de topic "/house/switch1" gebruikt is, zal de esp zijn taak uitvoeren. De topic bevat een waarde, als deze waarde "1" of "hoog" is, zal hij de schakelaar doen uitschakelen

(relais gebruikt omgekeerde logica). Wanneer de schakelaar in die toestand is veranderdt, zal de esp een confirmatie geven aan de broker dat het proces goed verlopen is. Enkel en alleen als deze confirmatie gegeven is, zal de status op het beginscherm veranderen. De topic "/house/switchConfirm1/" wordt hiervoor gebruikt

```
if (topicStr == "/house/switch1/")
{
    if(payload[0] == '1'){
        digitalWrite(switchPin1, LOW);
        client.publish("/house/switchConfirm1/", "1");
    }

    else if (payload[0] == '0'){
        digitalWrite(switchPin1, HIGH);
        client.publish("/house/switchConfirm1/", "0");
    }
}
```

Afbeelding 14: Arduino code switch

```
switch 1:
- platform: mqtt
  name: "MQTT Switch 1"
  state_topic: "/house/switchConfirm1/"
  command_topic: "/house/switch1/"
  payload_on: "1"
  payload_off: "0"
  qos: 0
  retain: true
```

Afbeelding 14: configuration switch

3.3.3 Multisensor

Omdat je met sensoren heel leuke automatisaties kan maken heb ik een multisensor vooraan op mijn lijstje gezet. Ik heb vele opties bekeken. De multisensor van Aeotec die ook gebruik maakt van het Z-wave communicatieplatform was een goede optie, maar vanwege de kostprijs van 60 euro was deze toch niet zo geschikt. Ik heb nog onderzoek gedaan naar multisensoren die je kan aankopen. Uiteindelijk kwam ik terecht bij een DIY-multisensor die gebruik maakt van MQTT. Er is een stap voor stap tutorial te vinden op Github

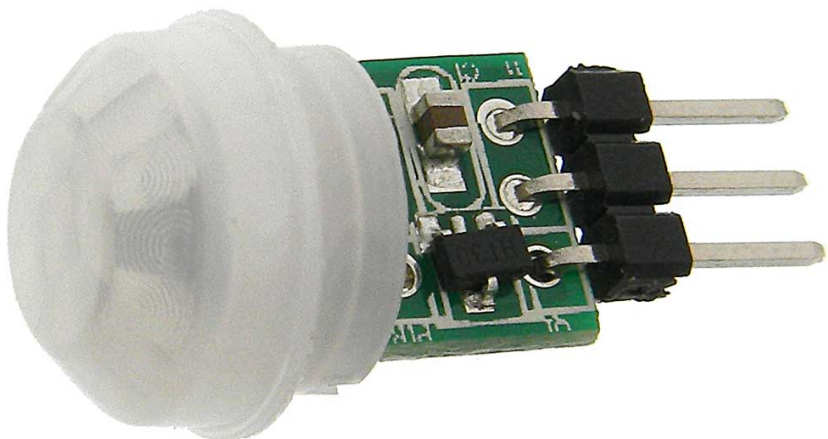
(<https://github.com/bruhaautomation/ESP-MQTT-JSON-Multisensor>). Er is zelfs een video beschikbaar die alle stappen één voor één overloopt. De multisensor implementeert een temperatuursensor, bewegingssensor, lichtsensor en een luchtvochtigheidssensor in één apparaat dat je verbindt met de Raspberry Pi. Het is geen simpel proces, je moet zelfs de files van de library aanpassen om deze compatibel te maken met de esp. Je moet ook een extra library installeren, wat niet vermeld is in de tutorial. Zonder deze library zal je de code niet kunnen uploaden. Dit was één van de punten waar ik vast liep en niet verder kon. Je kan alle componenten makkelijk verbinden zonder dat er extra hardware nodig is. Ik heb nog niet alle sensoren kunnen uitlezen. Ik heb enkel de bewegingssensor en de lichtsensor

kunnen aansluiten omdat de andere componenten na een bestelling op Aliexpress nog niet aangekomen zijn.

Voor de

bewegingsdetectie

gebruik ik een pir-module. Ik heb gekozen voor een module omdat deze



Afbeelding 17: mini pir sensor AM312

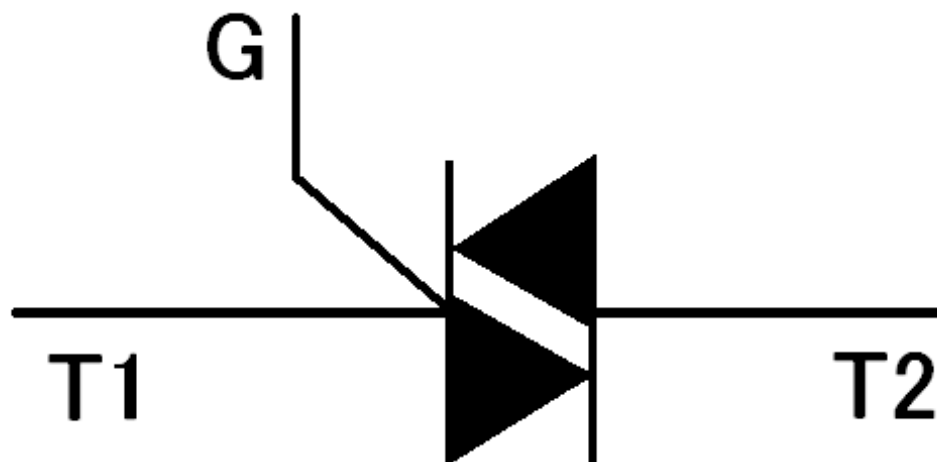
veel betrouwbaarder is. Door de signalen die de esp uitzendt en ontvangt gaat de pir-sensor vaak storingen meten, de control module zorgt ervoor dat er een juist signaal aan de esp wordt gegeven. De **passive infrared** sensor meet het infrarood licht dat verschillende voorwerpen uitstralen. Het is een passieve sensor omdat de sensor geen straling genereert om zijn detectie te kunnen doen. Het menselijk lichaam heeft een constante warmte, dus straalt je altijd een bepaalde hoeveelheid infraroodstraling uit. De sensor meet echter niet de warmte, hij meet enkel het verschil in straling. Wanneer een voorwerp deze pir-sensor passeert zal er een verandering in de hoeveelheid straling zijn. De waarneming van de kamertemperatuur zal veranderen in een grotere waarde omdat het menselijk lichaam een hogere temperatuur heeft. Deze verandering zal dan beweging aantonen.

Er zijn veel verschillende opties voor lichtdetectie. Ik heb gekozen voor een photoresistor of light-dependent resistor. Zoals de naam al doet vermoeden, is dit een weerstand die in functie van de hoeveelheid licht zijn weerstandwaarde zal veranderen. De ldr is gemaakt uit een halfgeleidend materiaal met een hoge weerstandswaarde. Wanneer het materiaal blootgesteld wordt aan licht, zal de energie van de fotonen geabsorbeerd worden door de elektronen. Hierdoor zal de ldr een lagere weerstand krijgen. Het grote voordeel van de ldr is het aansluiten en het uitlezen. Je kan een ldr aansluiten zoals een potentiometer en zo een analoge waarde uitlezen. Er is enkel 1 analoge poort aanwezig op de esp dus zal je enkel 1 analoge sensor kunnen uitlezen.

3.3.4 Dimmer

Nadat ik een laboproef over de triac had afgerond tijdens de lessen vermogenselektronica, kreeg ik de inspiratie om een dimmer te maken die bestuurbaar is met Home Assistant. Ik begon dus met de triac. Deze kon ik al via een Arduino aansturen en zo een lamp dimmen. Maar omdat ik deze via een esp wou besturen moest ik een programma schrijven dat MQTT-functionaliteit implementeert. Ik heb de standaard "MQTT Brightness" code gebruikt en deze aangepast. Deze code was bedoeld om een led via pwm aan te kunnen sturen, maar omdat een triac op een compleet ander concept werkt moest ik de code herwerken. Voor de MQTT-communicatie zijn er vier topics nodig: voor het aan-en uitschakelen, het dimmen en de confirmatie van deze twee variabelen.

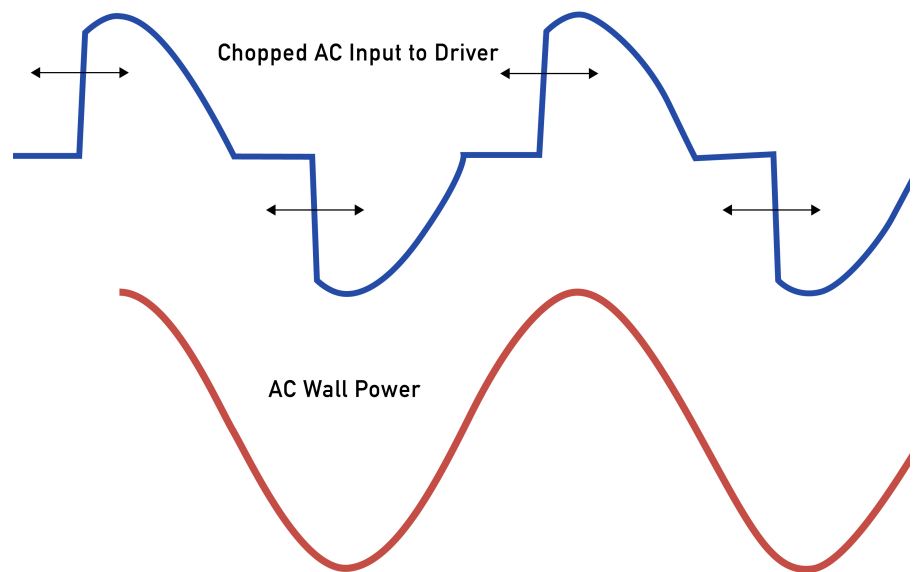
Wanneer je een lamp op een hoge wisselspanning gaat aansturen, is veiligheid natuurlijk de prioriteit van de opstelling. Ik gebruik twee opto-isolatoren om de stuurkring en de vermogenkring



Afbeelding 18: symbool van de triac

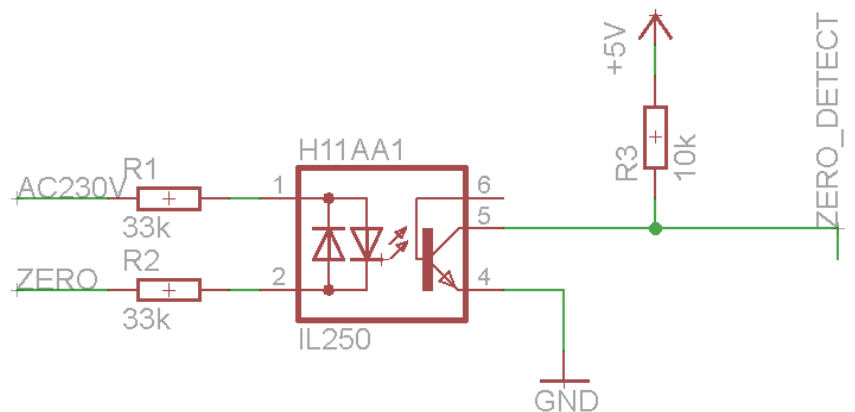
galvanisch te scheiden. Om de lamp te kunnen dimmen gebruik ik een triac. De triac is een antiparallelschakeling van twee thyristoren. Er is echter maar één gemeenschappelijke stuurlijn(gate) aanwezig. De triac zal zowel positieve als negatieve spanningen kunnen schakelen. Deze constructie is dus een wisselstroomschakelaar. De benamingen van de thyristor, anode en kathode zal dus vervangen worden door T1 en T2. Eenmaal de triac in geleiding gebracht is, zal de triac in geleiding blijven. De triac zal enkel doven wanneer de stroom door de triac onder de

houdstroom komt. Door deze karakteristiek kunnen we een wisselspanning gaan aansnijden. We sturen de triac aan door een kleine spanning op de gate aan te sluiten. Wanneer er een kleine stroom vloeit zal de triac ontsteken en geleiden. Als de momentele waarde van de wisselspanning daalt tot onder de houdspanning zal de triac weer sperren. Hierdoor kunnen we elke halve periode aansnijden en zo de gemiddelde spanning regelen.



Afbeelding 15: aangesnede wisselspanning

Om deze aansnijding te kunnen verwezenlijken moeten we natuurlijk weten waar de nuldoorgangspunten zijn. De nuldoorgangdetectie gebeurt door een optocoupler met een phototransistor als



Afbeelding 20: nuldoorgangsdetectie met H11AA1 optocoupler

uitgang. Door enkele weerstanden toe te voegen kunnen we een kleine puls verkrijgen wanneer de spanning door zijn nulpunt gaat. Deze kleine puls wordt dan verbonden met een interruptieingang op de esp. Wanneer er een puls wordt gedetecteerd, zal de esp een uitgangspuls geven van 200

microseconden. De waarde X wordt gebruikt om te vermijden dat er een kleine stroom gaat vloeien wanneer de lamp volledig gedimd is.

```
void nuldoorgang() {
  val = map(m_light_brightness, 0, 255, 9000, 1);

  if (x == 1) {
    delayMicroseconds(val);
    digitalWrite(4, HIGH);
    delayMicroseconds(200);
    digitalWrite(4, LOW);
  }

  if (x == 0) {
    digitalWrite(4, LOW);
  }

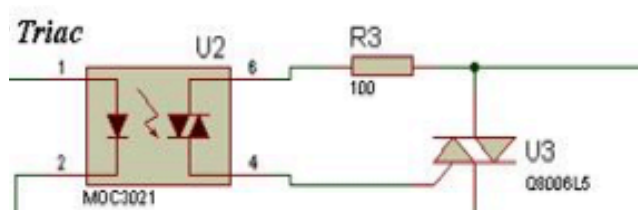
  if (val > 8500) {
    x = 0;
  }

  Serial.println(val);
}
```

Afbeelding 21: hoekbepaling voor fase aansnijding

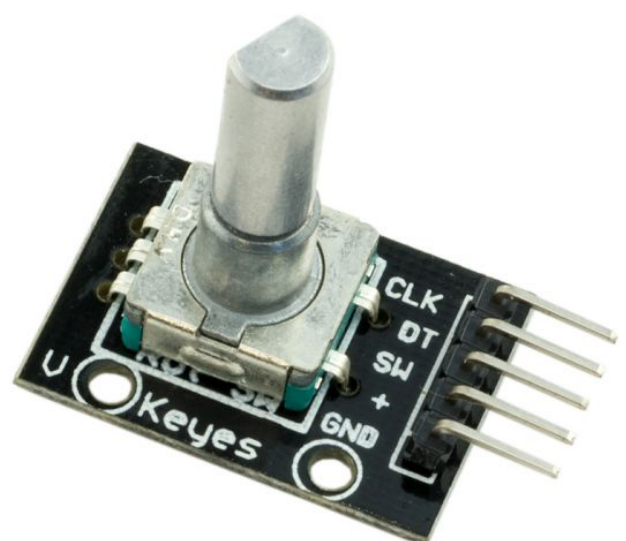
Omdat we elke halve periode kunnen aansnijden die 10 milliseconden duurt, moeten we de stuurpuls binnen een tijdspanne van 10 milliseconden geven. Door een kleinere of grotere vertraging te kiezen kunnen we de ontsteekhoek bepalen. Deze uitgangspuls wordt weer aangesloten aan een optocoupler, nu is het een optocoupler met een triac-uitgang.

Deze optocoupler kan enkel 1 ampère sturen aan zijn uitgang. Het is natuurlijk de bedoeling dat we een groot vermogen zoals een lamp gaan kunnen besturen. We gebruiken het signaal van de optocoupler om hiermee een stuurpuls te geven aan een triac die een groter vermogen kan schakelen.



Afbeelding 22: schema aansturing vermogenstriac

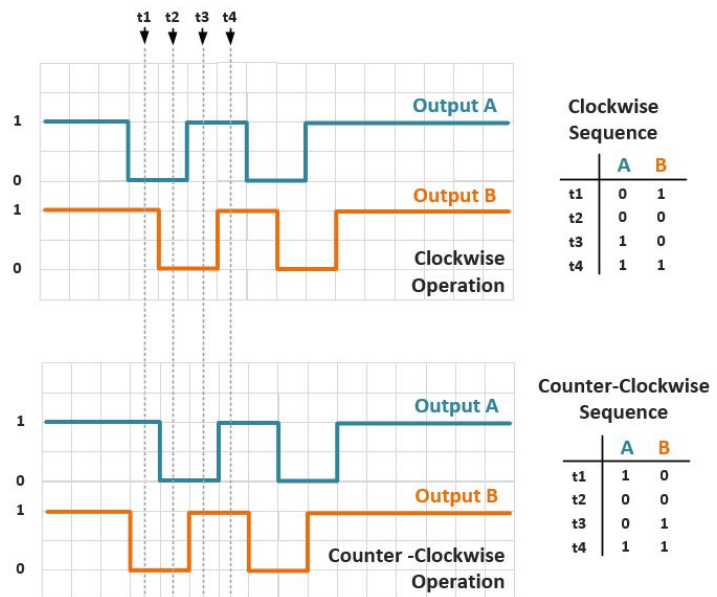
Ik heb al eerder aangehaald dat de besturing niet alleen via de software mogelijk moet zijn, je moet het ook handmatig kunnen besturen. Bij een dimmer zijn de opties natuurlijk beperkt. De lamp moet aan-en uitgeschakeld worden en moet gedimd kunnen worden. Conventionele dimmers gebruiken een



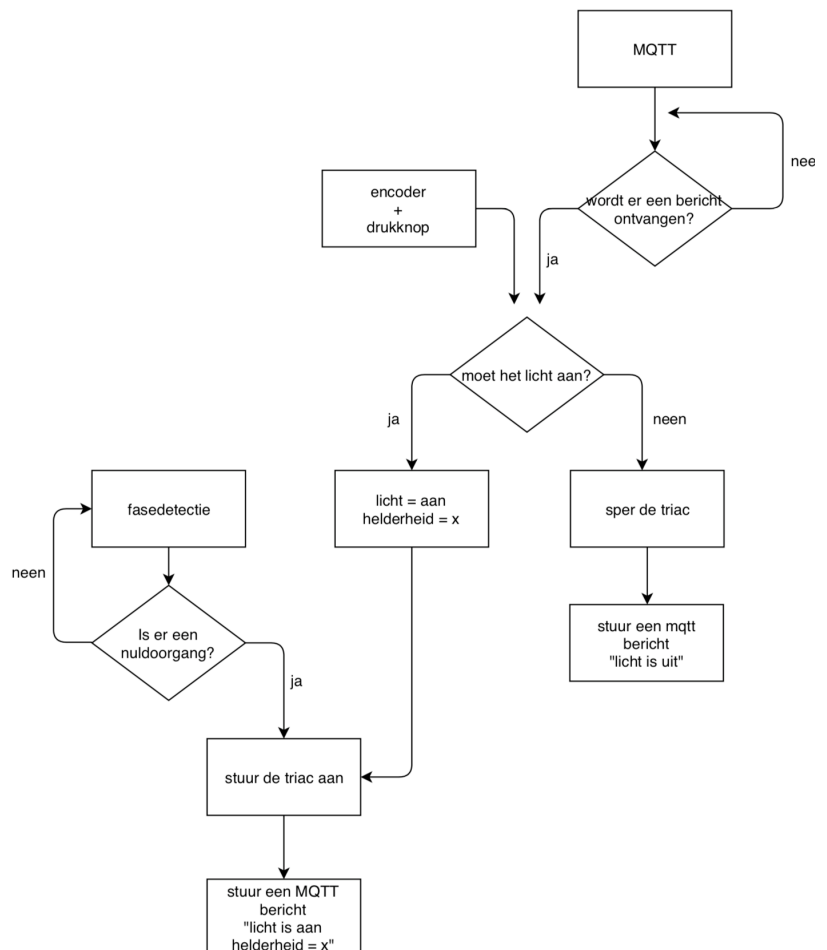
Afbeelding 23: encodermodule

potentiometer. Bij een dimmer die je via twee besturingsmethodes kan aansturen is dit natuurlijk niet mogelijk door de vaste stand van de potentiometer. Daarom heb ik gekozen voor een encoder met een ingebouwde drukknop. De encoder heeft een roterende as. Wanneer er aan die as gedraaid wordt, genereert de encoder pulsen. Denk hier bijvoorbeeld aan een draaiknop in de auto. Met deze pulstrein kan je bepalen in welke richting er gedraaid wordt.

De dimmer is nog niet volledig af, ik heb nog veel ideeën die ik wil implementeren. Zoals drukknoppen die te vinden zijn op de behuizing waarmee je allerlei andere apparaten kunt aansturen. Zo zal je bijvoorbeeld een scène kunnen activeren die meerdere lichten zal aansturen.

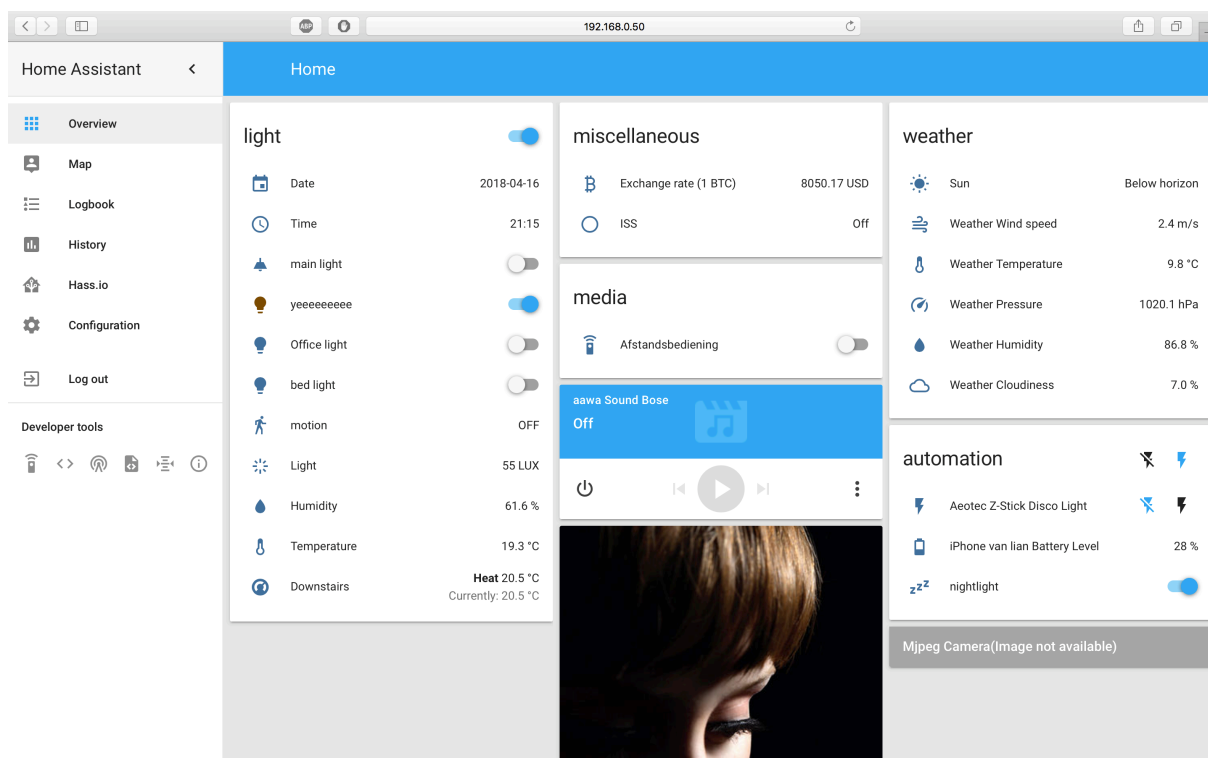


Afbeelding 24: pulstrein wanneer er in beide richtingen gedraaid wordt



Afbeelding 25: blokschema dimme

3.4 Andere Widgets



Afbeelding 26: voorpagina Home Assisant

Er zijn nog honderden andere widgets beschikbaar waarvoor je zelf geen hardware nodig hebt. Zoals je kan zien heb ik enkele widgets op de voorpagina ter beschikking. De data van het weer zijn beschikbaar via de software van Home Assistant. Er moeten enkele regels toegevoegd worden in de configuratiefile. Ik gebruik de data van een Noorse website die globaal informatie verzamelt. Om Spotify te integreren heb je een beetje meer werk. Ten eerste moet je account geregistreerd worden als developer. Daarna moet je een applicatie maken op de developer website. Wanneer de applicatie aangemaakt is, zal je een "Client ID" en een "Client Secret" toegewezen krijgen. Deze gegevens moeten in de configuratiefile verwerkt worden. Als alles werkt, zal er een mediaspeler tevoorschijn komen. Je kan bijvoorbeeld via je gsm muziek beginnen te streamen op je computer.

Er zijn ook veel personalisatie opties beschikbaar, je kan alle namen veranderen. Er is ook een optie om andere symbolen te kiezen. Via de site van Home Assistant zijn er letterlijk honderden verschillende symbolen beschikbaar.

We hebben recent ook een Sonos Playbase aangekocht. Het eerste dat natuurlijk in mij opkwam was de optie om deze ook te besturen



Afbeelding 27: Sonos Playbase

via Home Assistant. Het configureren was vanzelfsprekend. Op de website van Home Assistant kan je een gedetailleerde handleiding vinden. Je moet enkel het IP-adress vinden van het apparaat. Dit is makkelijk te doen met een gratis app zoals Fing(<https://www.fing.io>). Wanneer het Sonos-apparaat verbonden is, kan je zelfs kiezen uit verschillende afspeellijsten die je wilt afspelen.

3.4.1 Device tracker

Er is ook een device tracker beschikbaar in Home Assistant. Via iCloud kan je de locatie van verschillende Apple-apparaten bepalen die door hetzelfde familieaccount verbonden zijn. Dit is natuurlijk uit te schakelen door de individuele gebruikers van het familieaccount zodat de privacy van de gebruikers bewaard blijft. Het enige grote nadeel is dat je apparaat constant zijn locatie zal bepalen, en hierdoor zal je batterij veel sneller leeglopen. Het enige dat nodig is in de configuratiefile zijn de logingegevens van je iCloud account. Na wat updates en dagelijks gebruik

heb ik de device tracker uiteindelijk uitgeschakeld vanwege het grote batterijgebruik.

Omdat ik toch een device tracker wil hebben zonder een extreem groot batterijverbruik, ben ik op zoek gegaan om de website van Home Assistant. Ik heb uiteindelijk een Bluetooth tracker gevonden. Je zal eerst de Bluetooth add-on moeten installeren, deze zal de Bluetooth functionaliteit van de Raspberry Pi

3b activeren. Er is geen extra configuratie nodig voor deze add-on. Er is wel een kleine configuratie nodig voor de tracker zelf. De Raspberry Pi zal

```
# Example configuration.yaml entry
device_tracker:
  - platform: bluetooth_tracker
```

Afbeelding 28: cofiguratie van de tracker

verschillende bluetoothsignalen detecteren. Hierdoor moet je bluetooth altijd geactiveerd zijn op het apparaat dat je wilt tracken. Mijn bluetooth is altijd geactiveerd, het batterijgebruik van mijn smartphone zal dus niet beïnvloed worden. Deze tracker zal niet je exacte locatie kunnen bepalen. De Raspberry Pi zal enkel je aanwezigheid kunnen detecteren als je bluetoothsignaal waarneembaar is door de bluetoothmodule.

3.4.2 Nest

Er is ook een optie om Nest-producten te verbinden met Home Assistant. Nest is een home automation bedrijf dat recent is overgenomen door Google, het bedrijf maakt



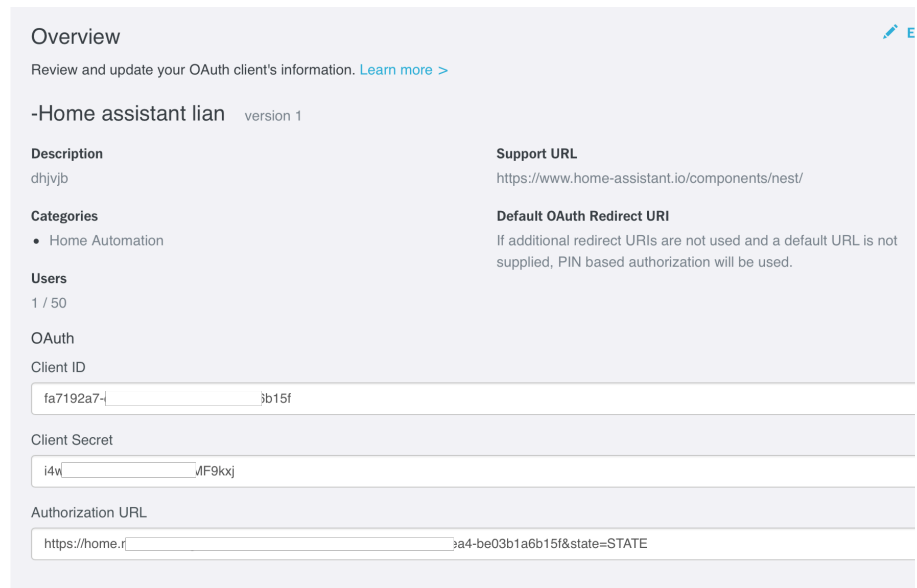
Afbeelding 29: Nest Thermostaat

vooral
beveiligingsproducten
zoals camera's,
deurbellen en
bewegingsmelders.
Ik heb enkel de "Nest
Learning Thermostat"
gebruikt. Deze slimme
thermostaat kan door
middel van beweging
een
verwarmingsschema

opbouwen naargelang de activiteit die aanwezig is in het huis. De
thermostaat is natuurlijk ook te verbinden met Home Assistant. Het is
echter een moeilijke configuratie. Ten eerste moet je een account maken
op de "Nest Developer" site, waar je een uitgebreide vragenlijst moet
invullen. Daarna moet je een nieuw "product" aanmaken. Je zal zoals in
de Spotify-configuratie een "client ID" en een "client secret" krijgen.
Wanneer je Home Assistant weer opstart nadat je de configuratie hebt
aangepast, zal je een beveiligingscode moeten invullen die tijdelijk te
vinden is op de "Nest developer" site.

3.5 Apple Homekit

Voice control is zeker een leuke functie in
een home automation systeem. Er is
geen betere partytrick dan met je eigen
stem een leuk sfeerlichtje en een beetje
muziek opzetten. Er zijn drie grote voice
control systemen op de markt:



The screenshot shows the 'Overview' page for an OAuth client named '-Home assistant lian' (version 1). It includes fields for Description, Categories (Home Automation), Users (1 / 50), OAuth Client ID (fa7192a7-...), Client Secret (i4w...), and Authorization URL (https://home.r...). It also lists the Support URL and Default OAuth Redirect URI.

Overview	
Review and update your OAuth client's information. Learn more >	
-Home assistant lian version 1	
Description dhjvjvb	Support URL https://www.home-assistant.io/components/nest/
Categories • Home Automation	Default OAuth Redirect URI If additional redirect URIs are not used and a default URL is not supplied, PIN based authorization will be used.
Users 1 / 50	
OAuth	
Client ID fa7192a7-...b15f	
Client Secret i4w...jF9kxj	
Authorization URL https://home.r...ja4-be03b1a6b15f&state=STATE	

Afbeelding 30: Nest Developer pagina



Afbeelding 31: Homekit App

Google Assistant, Amazon Alexa en natuurlijk Siri van Apple. Home Assistant ondersteunt alle drie de opties. Ik heb gekozen voor Siri omdat deze in mijn situatie de meest toegankelijke optie is. Ik ben een gebruiker van het IOS-platform, daardoor heb ik geen bijkomende hardware nodig zoals de Google Home of de Amazon Echo. De tamelijk goedkope prijs van deze smartspeakers is natuurlijk een groot pluspunt. Je kan niet zomaar elk apparaat verbinden met Homekit, het moet een compatibel apparaat zijn. Home Assistant is spijtig genoeg niet ondersteund. Natuurlijk zijn er altijd wel manieren om dit te omzeilen.

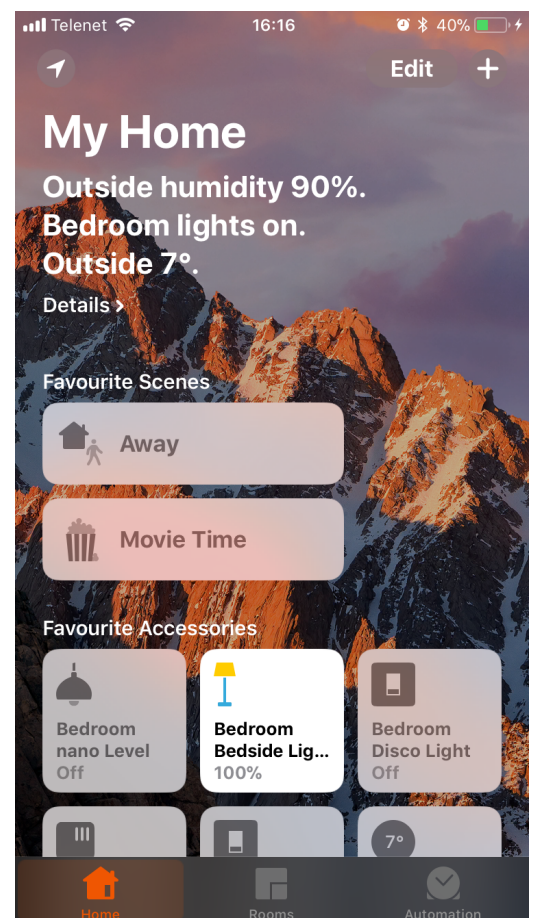
Ik gebruik de add-on "Homebridge" om al de apparaten te besturen via Apple Homekit. Homebridge emuleert een Philips Hue apparaat. Philips Hue is een productenlijn van allerlei verschillende lampen die bestuurbaar zijn via een draadloze verbinding. Deze lichten zijn wel compatibel met Homekit, dit was de inspiratie voor de ontwerpers van deze add-on. Alle Philips Hue lampen zijn verbonden met een zogenaamde HueBridge, deze centrale controller bestuurt alle lichten. Homebridge emuleert deze zogenaamde "brug". Eenmaal de add-on geïnstalleerd is, zal er een extra config file verschijnen. In

```

{
  "bridge": {
    "name": "Home Assistant",
    "username": "B8:27:EB:81:E1:B9",
    "port": 51826,
    "pin": "169-42-325"
  },
  "description": "Homebridge for Home Assistant",
  "accessories": [],
  "platforms": [
    {
      "platform": "HomeAssistant",
      "name": "HomeAssistant",
      "host": "http://hassio/homeassistant",
      "password": "",
      "default_visibility": "hidden",
      "supported_types": [
        "automation",
        "binary_sensor",
        "climate",
        "cover",
        "device_tracker",
        "fan",
        "group",
        "input_boolean",
        "light",
        "lock",
        "media_player",
        "remote",
        "scene",
        "script",
        "sensor",
        "switch",
        "vacuum"
      ]
    },
    {
      "logging": true,
      "verify_ssl": true
    }
  ]
}

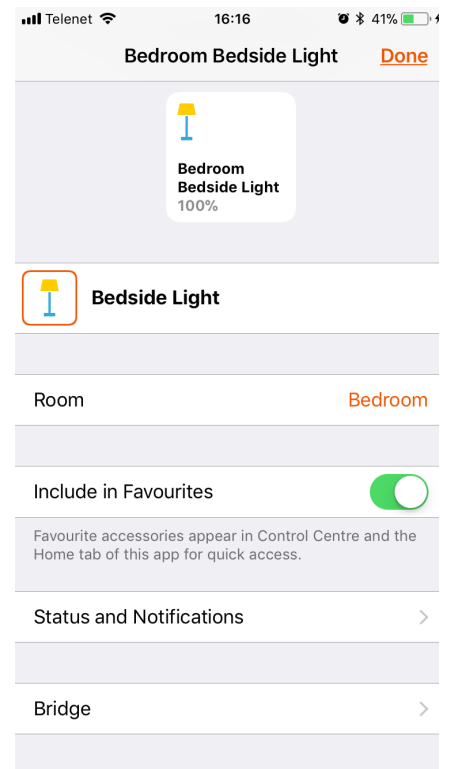
```

**Afbeelding 32: config file
Homebridge**



**Afbeelding 33: Homekit
voorpagina**

deze config file zal je zelf een pincode kunnen kiezen die je later in je Homekit app moet intikken. Je kan hier ook alle apparaten verbergen voor de Homekit app. Dit is handig zodat je later zelf kan kiezen welke apparaten je zichtbaar wilt maken, zo hou je alles ordelijk. Wanneer de add-on opgestart is, zal je op de Homekit app een apparaat kunnen toevoegen. Eerst moet je de pincode intikken die je zelf hebt gekozen. De applicatie zal een waarschuwing geven dat de Home Assistant component niet erkend is door Homekit. De waarschuwing verwaarloos je. Voor je alles kan besturen moet je eerst alle componenten die je zichtbaar hebt gemaakt configureren in Homekit. Dit is een snel proces. Je kan hier bijvoorbeeld een naam kiezen, een icoontje instellen, zeggen in welke kamer het behoort, ... Homekit biedt niet de volledige functionaliteit die in Home Assistant beschikbaar is. Dit geldt vooral bij de ingewikkeldere componenten zoals mediaspelers. Alle lichtschakelaars en sensors behouden wel dezelfde functionaliteit. Je kan ook scenes maken, met de druk op één knop ga je van een donkere kamer naar een gezellige kamer waar alle lichten met de juiste helderheid branden. Ga je het huis uit? Druk dan even op de knop "afwezig" en alles schakelt vanzelf uit. Je hebt natuurlijk ook de leuke functionaliteit van Siri. Je kan niet alleen lampen uitschakelen maar je kan bijvoorbeeld ook vragen of je lampen aan zijn. Het dimmen is ook mogelijk door een percentage te vermelden wanneer je de lampen inschakelt. Nu moet ik enkel mijn ouders overtuigen om een HomePod te kopen...



Afbeelding 34: configuratie apparaten in Homekit

3.6 Automatie

Automatische processen zijn natuurlijk ontworpen om het leven efficiënter te maken. Wat is het punt allerlei dure apparaten aan te kopen zonder zaken te automatiseren?

Een automatie bestaat uit 3 grote delen: een "trigger", een "condition" en een "action". De trigger zal de automatie starten, in dit voorbeeld zal de automatie starten één uur voor de zon ondergaat. De condition is een voorwaarde. Als aan deze voorwaarde niet voldaan is, zal de automatie niet uitgevoerd worden. Er kan maximaal één trigger bestaan, maar er kunnen wel meerdere conditions aanwezig zijn. In dit voorbeeld is de voorwaarde dat de gebruiker thuis moet zijn. Eenmaal de automation getriggerd en aan de voorwaarde voldaan is, zullen de lichten aangaan. Door de automatisaties, zijn de mogelijkheden onbeperkt. Je kan alles aanpassen naargelang je voorkeur. Ik heb bijvoorbeeld een automatie geconfigureerd waarbij het licht 's morgens aangaat. Wanneer het 6:32 is, zal de automatie triggeren. De automatie zal enkel uitgevoerd worden voor zonsopgang. Als aan de voorwaarde voldaan is, zal de lichtintensiteit op een tijd van 15 minuten toenemen. In de hoop dat ik dan tijdig opsta om naar school te vertrekken.

```
automation:
  alias: Turn on the lights when the sun sets
  trigger:
    platform: sun
    event: sunset
    offset: "-01:00:00"
  condition:
    condition: state
    entity_id: group.all_devices
    state: 'home'
  action:
    service: light.turn_on
```

Afbeelding 35: configuratiefile automatie

```
automation:
- alias: 'opstaan'
  trigger:
    - platform: time
      at: "06:32:00"
  condition:
    - condition: sun
      before: sunrise
  action:
    service: light.turn_on
    entity_id: light.nano_level
    data:
      # 900 seconden = 15 minuten
      transition: 900
```

Afbeelding 36: automation "opstaan"

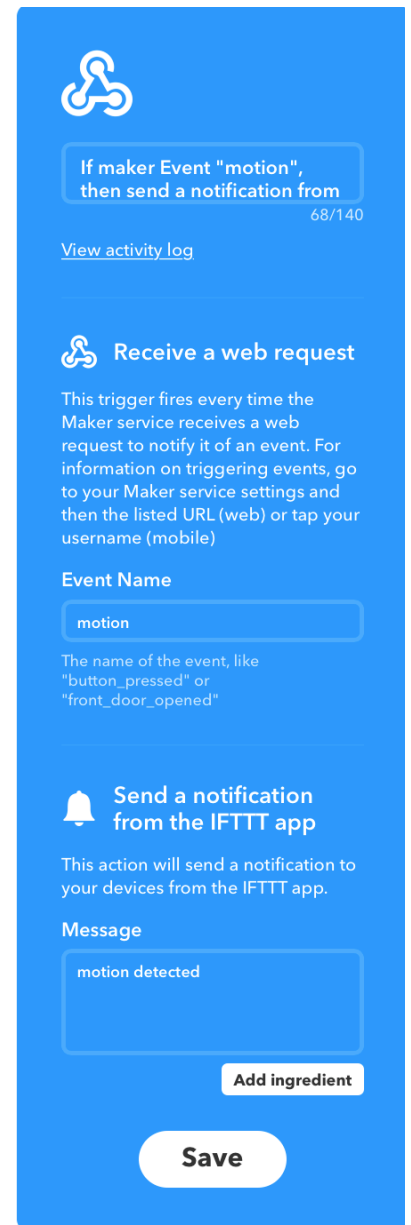
3.6.1 IFTTT

If This Than That is een web service gemaakt voor het linken van verschillende acties. Je hebt altijd een trigger en een actie. Deze trigger kan van verschillende bronnen komen.

Je kan dus ook een IFTTT “applet” triggeren door een gebeurtenis in Home Assistant. Ik ben dus beginnen experimenteren met de mogelijkheden. Deze applet wordt getriggerd uit Home Assistant wanneer er een bericht met de data “motion” wordt ontvangen. Hierna zal IFTTT een notificatie sturen naar je smartphone. Omdat dit ook een sms-bericht kan zijn, hoeft je dataverkeer dus niet constant op te staan. Hiervoor heb je ook een kleine automation nodig die dan de data naar IFTTT stuurt.

Je kan ook een oproep maken via IFTTT. Zo heb ik een automatie gemaakt die me belt wanneer ik mijn smartphone verloren ben. Deze automatie activeer ik

dan op Home Assistant. Wanneer ik dit telefoontje opneem, wordt er vriendelijk vermeld dat ik hem terug heb gevonden.



Afbeelding 37: IFTTT applet

```
automation:
- alias: motion allarm
  trigger:
    - platform: state
      entity_id: sensor.motion
      to: 'ON'
  condition:
    - condition: state
      entity_id: device_tracker.iphone_van_lian
      state: 'away'
  action:
    - service: ifttt.trigger
      data: {"event":"motion"}
```

Afbeelding 38: IFTTT automation in de configuration file

3.7 Ondervonden problemen

Tijdens het samenstellen van mijn gip heb ik verschillende problemen ondervonden. De meeste ondervonden problemen heb ik al aangehaald in de vorige hoofdstukken.

Eén probleem dat constant voorkwam, was het crachen van de Raspberry Pi. Hieruit heb ik geleerd dat het maken van een back-up je uren onnuttig werk kan besparen. Er is een ingebouwde tool aanwezig in Home Assistant waarmee je makkelijk een back-up kan maken. Je mag wel niet vergeten deze back-up uit de files te kopiëren en dan goed te bewaren.

De Raspberry Pi was ook heel onbetrouwbaar, want deze is meerdere keren gecrasht op willekeurige momenten. Ik had letterlijk alles geprobeerd om dit te verhelpen. Ik kwam tot de conclusie dat het gebruikte SD-kaartje te traag was waardoor de taken niet kon uitgevoerd. Ik heb een andere SD-kaart gebruikt om dit probleem op te lossen.

Wanneer ik de zelfgemaakte dimmer had afgewerkt, ben ik deze dan ook intensief beginnen gebruiken buiten de schooluren. De lamp schakelde op willekeurige momenten uit. Ik had geen enkel idee wat de oorzaak hiervan kon zijn. Tijdens de labo-uren ben ik beginnen debuggen. Via de seriële monitor heb ik een resetcode kunnen vinden.

Na wat onderzoek bleek dit de zogenaamde “watchdog” te zijn. De watchdog zorgt ervoor dat het systeem zich automatisch reset wanneer het programma vastloopt. Het lopende programma stuurt een signaal naar deze watchdog om te confirmeren dat het programma nog operationeel is. Hierdoor kunnen enkele lange acties niet uitgevoerd worden.

De interrupt is de oorzaak van dit probleem. Wanneer er een interrupt getriggerd wordt, zal de faseaansnijding uitgevoerd worden. Elke 10 milliseconden wordt deze interrupt getriggerd. Maar omdat de faseaansnijding maximaal 10 milliseconden kan duren is er geen tijd om een signaal naar de watchdog te sturen voor de volgende interrupt wordt getriggerd. Dit gebeurt natuurlijk niet altijd. De watchdog moet elke seconde een signaal krijgen. Het resetten kan dus willekeurig gebeuren wanneer er even geen signaal is. Door verschillende omstandigheden kan dit enkelen minuten of zelfs enkele uren duren.

4 Besluit

Om het levensgemak te verbeteren in een thuisomgeving heb ik open source home automation software gebruikt. Hierdoor is er geen extensieve aanpassing nodig van het bestaande thuisnetwerk. Er wordt veel gebruik gemaakt door draadloze communicatie om zo sensoren en actuatoren met elkaar te verbinden.

Uiteindelijk heb ik erg genoten van het sleutelen aan deze proef, niet alleen tijdens de schooluren maar vooral in mijn vrije tijd. Ik heb tijdens deze uren een uitgebreid netwerk van geconnecteerde actuatoren en sensoren kunnen verwezenlijken. Mijn positieve houding tegenover deze proef heeft me geholpen om zo gemotiveerd mogelijk te werken.

Het configureren was een leuke tijdsbesteding met een groot gevoel van voldoening. Omdat ik het project zelf gefinancierd heb, beleefde ik er thuis veel plezier aan. Wanneer ik het licht activeer via Google Assistant of door een zelfgemaakte module word ik steeds herinnerd aan mijn eigen werk dat hiervoor nodig was.

Het project was zeker niet eenvoudig en vanzelfsprekend. Er waren veel momenten waar ik geen progressie maakte. Dit is natuurlijk geen leuk gevoel, maar wanneer je uiteindelijk het juiste resultaat krijgt kan dit je dag zo veel beter maken. Het tijd dat ik besteed aan deze proef ging soms ook ten koste van ander schoolwerk met alle gevolgen vandien.

Het is zeker een geslaagd eindproduct. Mijn kamer is omgetoverd in een slimme omgeving waarvan je elke dag kan genieten. Er zijn nog eindeloze mogelijkheden over. Ik ga zeker niet stoppen met het ontdekken en uitvoeren van deze mogelijkheden. Wanneer je eenmaal een slimme omgeving hebt gecreëerd is er altijd een aspect dat je wilt verbeteren. Dit project gaat zeker en vast niet in de kast belanden om stof te verzamelen.

5 Bibliografie

ADAFRUIT, A, cdn-shop.adafruit.com, internet, 1 juni 2015.
(https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266_Datasheet_EN_v4.3.pdf)

AEOTEC, A, aeotec.freshdesk.com, internet.
(<https://aeotec.freshdesk.com/support/solutions/6000084017>)

ALEH, A, github.com, internet, 11 maart 2017
(<https://github.com/aleh/AAL-EC11>)

AVENUE33, A, forum.arduino.com, internet, 21 september 2016.
(<https://forum.arduino.cc/index.php?topic=425195.0>)

BOURNS, B, ret.hu, internet, september 2002.
(<https://www.ret.hu/media/product/6874/426254/tic226.pdf>)

BRUH AUTOMATION, BA, [youtube.com](https://www.youtube.com), internet, 13 augustus 2017.
(<https://www.youtube.com/channel/UCLecVrux63S6aYiErxdy4w>)

CORBANMAILLOUX, C, github.com, internet, 13 december 2017.
(<https://github.com/corbanmailloux/esp-mqtt-rgb-led>)

FARNELL, F, [farnell.com](http://www.farnell.com), internet, september 2012.
(<http://www.farnell.com/datasheets/1708247.pdf>)

FARNELL, F, [farnell.com](http://www.farnell.com), internet, november 2008.
(http://www.farnell.com/datasheets/2309902.pdf?_ga=2.56747818.1801992972.1521482970-1380377901.1519144176&_gac=1.247130672.1521482970.EAIaIQobChMIz6jI2_342QIVUImyCh1sFAH4EAAyASAAEgIA-fD_BwE)

HI LINK, HL, [hlktech.net](http://www.hlktech.net), internet.
(http://www.hlktech.net/product_detail.php?ProId=54)

FRENCK, F, community.home-assistant.io, internet, 13 mei 2018.
(<https://community.home-assistant.io/t/community-hass-io-add-on-homebridge/33803>)

HOME ASSISTANT, HA, github.com, internet, 18 februari 2018.
(<https://github.com/home-assistant/homebridge-homeassistant>)

HOME ASSISTANT, HA, home-assistant.io, internet, 11 mei 2018.
(<https://home-assistant.io>)

MAATTDIY, M, [github.com](https://github.com/maattdiy/home-assistant-config), internet, 2 maart 2018.
(<https://github.com/maattdiy/home-assistant-config>)

MERTENATS, M, [github.com](https://github.com/mertenats/Open-Home-Automation), internet, 10 mei 2018.
(<https://github.com/mertenats/Open-Home-Automation>)

NODEMCU, N, [nodemcu.readthedocs.io](https://nodemcu.readthedocs.io/en/master/), internet.
(<https://nodemcu.readthedocs.io/en/master/>)

RANDOM NERD TUTORIALS, RNT, randomnerdtutorials.com, internet.
(<http://randomnerdtutorials.com/10-diy-wifi-rgb-led-mood-light-with-esp8266-step-by-step/>)

RASPBERRY PI, RP, [raspberrypi.org](https://www.raspberrypi.org), internet,
(<https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>)

RESINOS, R, docs.resin.io, internet.
(<https://docs.resin.io/deployment/network/2.0.0/>)

SPARKFUN, S, www.sparkfun.com, internet.
(<https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>)

SUNROM, S, sunrom.com, internet.
(<https://www.sunrom.com/p/micro-pir-motion-detection-sensor-am312>)

VISHAY, V, vishay.com, internet, 30 augustus 2016.
(<https://www.vishay.com/docs/83505/k3020p.pdf>)

VISHAY, V, vishay.com, internet, 23 februari 2011.
(<https://www.vishay.com/docs/83608/h11aa1.pdf>)

VISHAY, V, vishay.com, internet, 11 juni 2013.
(<https://www.vishay.com/docs/83629/il420.pdf>)

6 Bijlage

6.1 configuration.yaml

```
homeassistant:
  # Name of the location where Home Assistant is running
  name: Home
  # Location required to calculate the time the sun rises and sets
  latitude: 51.24285187577013
  longitude: 5.139476833869833
  # Impacts weather/sunrise data (altitude above sea level in meters)
  elevation: 28
  # metric for Metric, imperial for Imperial
  unit_system: metric
  # Pick yours from here:
  # http://en.wikipedia.org/wiki/List_of_tz_database_time_zones
  time_zone: Europe/Brussels
  # Customization file
  customize: !include customize.yaml

# Show links to resources in log and frontend
introduction:

#####
##Goolge Home
#####
cloud:
  google_actions:
    filter:
      include_entities:
        - climate.downstairs
        - input_boolean.Nightlight
        - light.Bed_light
        - light.nano_level
        - light.office_light
        - light.Yeeeeeeeeee
        - sensor.humidity
        - sensor.light
        - sensor.temperature
      exclude_entities:
        - switch.mqtt_switch_1
        - switch.mqtt_switch_2
        - switch.mqtt_switch_3
        - switch.mqtt_switch_4
        - media_player.aawasoundtouch_speaker
        - device_tracker.lians_bedroom_speaker
        - media_player.lians_bedroom_speaker
        - switch.office_switch
        - media_player.spotify
        - switch.wallmote_switch
        - group.all_automations
        - group.all_devices
        - group.all_lights
        - group.all_remotes
        - group.all_scripts
        - group.all_switches
        - group.default_view
        - input_boolean.findphone
```

```

- group.media
- group.light
- group.automation
- input_boolean.lightturnon
- group.miscellaneous
- script.motionuitbedscript
- script.motionuityeescript
- switch.nano_switch
- switch.watchwinder
- group.weather
- light.yeelight_strip_7811dc670f5e

#####
##themes
#####
frontend:
  themes:
    teal:
      # Main Stuff #
      primary-color: "#1DE9B6" # Primary (most of the UI)
      primary-background-color: "#303030" # Primary background colour
(dialogs, e.t.c)
      secondary-background-color: "#303030" # Secondary background colour
(main UI background)
      paper-card-background-color: "#424242" # Card background colour
      paper-item-icon-color: "#1DE9B6" # Icon colour
      primary-text-color: "#FFFFFF" # Primary text colour
      secondary-text-color: "rgba(255, 255, 255, 0.7)" # Secondary text
colour
      disabled-text-color: "rgba(255, 255, 255, 0.5)" # Disabled text
colour
      divider-color: "rgba(255, 255, 255, 0.12)" # Divider colour
      paper-card-header-color: "#FFFFFF" # Card header text colour
      # Nav Menu #
      paper-listbox-background-color: "#424242" # Listbox background colour
      paper-listbox-color: "#FFFFFF" # Listbox text colour
      paper-grey-200: "#616161" # Listbox selected item background colour
      # Switches #
      paper-toggle-button-checked-ink-color: "#1DE9B6"
      paper-toggle-button-checked-button-color: "#1DE9B6"
      paper-toggle-button-checked-bar-color: "#1DE9B6"
      # Sliders #
      paper-slider-knob-color: "#1DE9B6"
      paper-slider-knob-start-color: "#1DE9B6"
      paper-slider-pin-color: "#1DE9B6"
      paper-slider-active-color: "#1DE9B6"
      paper-slider-secondary-color: "#33F2D8"
darkorange:
      primary-color: "#ff9800"
      light-primary-color: "#ffc947"
      dark-primary-color: "#c66900"
      text-primary-color: "#ffffff"
      primary-background-color: "#37464f"
      sidebar-text_-_background: "#62717b"
      paper-card-background-color: "#263137"
      paper-card-header-color: "var(--text-primary-color)"
      secondary-background-color: "var(--primary-background-color)"
      primary-text-color: "var(--text-primary-color)"
      secondary-text-color: "var(--primary-color)"
      paper-listbox-background-color: "var(--sidebar-text_-_background)"
      paper-toggle-button-checked-ink-color: "var(--dark-primary-color)"

```

```

paper-toggle-button-checked-button-color: "var(--dark-primary-color)"
paper-toggle-button-checked-bar-color: "var(--dark-primary-color)"
paper-slider-knob-color: "var(--primary-color)"
paper-slider-knob-start-color: "var(--primary-color)"
paper-slider-pin-color: "var(--primary-color)"
paper-slider-active-color: "var(--primary-color)"
paper-slider-secondary-color: "var(--light-primary-color)"
paper-slider-container-color: "var(--sidebar-text_-_background)"
paper-progress-secondary-color: "var(--sidebar-text_-_background)"
paper-item-selected_-_background-color: "var(--primary-color)"
paper-item-icon_-_color: "var(--text-primary-color)"
paper-item-icon-color: "var(--sidebar-text_-_background)"
paper-item-icon-active-color: "var(--primary-color)"
paper-grey-50: "var(--sidebar-text_-_background)"
paper-grey-200: "var(--primary-background-color)"
label-badge-background-color: "var(--sidebar-text_-_background)"
label-badge-text-color: "var(--text-primary-color)"
label-badge-red: "var(--primary-color)"
label-badge-blue: "var(--light-primary-color)"
label-badge-green: "var(--dark-primary-color)"
label-badge-yellow: "var(--text-primary-color)"
label-badge-grey: "var(--paper-grey-500)"
table-row-background-color: "var(--paper-card-background-color)"
table-row-alternative-background-color: "var(--sidebar-text_-_
background)"
darkred:
# Main colors that can be changed
dark-primary-color: "#c66900"
disabled-text-color: "#545454"
divider-color: "rgba(255, 255, 255, 0.12)"
light-primary-color: "#e06c6c"
paper-card-background-color: "#1d1d1d"
paper-grey-200: "#191919"
paper-item-icon-color: "#d3d3d3"
paper-listbox-background-color: "#202020"
primary-background-color: "#303030"
primary-color: "#d32f2f"
primary-text-color: "#cfcfcf"
secondary-background-color: "#131313"
sidebar-text_-_background: "#62717b"
# Colors based on variables, see above
paper-card-header-color: "var(--paper-item-icon-color)"
paper-item-icon-active-color: "var(--primary-color)"
paper-item-icon_-_color: "var(--primary-text-color)"
paper-listbox-color: "var(--primary-text-color)"
paper-grey-50: "var(--primary-text-color)"
paper-slider-active-color: "var(--primary-color)"
paper-slider-knob-color: "var(--primary-color)"
paper-slider-knob-start-color: "var(--primary-color)"
paper-slider-pin-color: "var(--primary-color)"
paper-slider-secondary-color: "var(--light-primary-color)"
paper-toggle-button-checked-ink-color: "var(--dark-primary-color)"
paper-toggle-button-checked-button-color: "var(--primary-color)"
paper-toggle-button-checked-bar-color: "var(--light-primary-color)"
paper-toggle-button-unchecked-bar-color: "var(--primary-text-color)"
secondary-text-color: "var(--primary-color)"
table-row-background-color: "var(--paper-card-background-color)"
table-row-alternative-background-color: "var(--sidebar-text_-_
background)"
darkcyan:
# Main colors that can be changed

```

```

primary-color: "#00bcd4"
disabled-text-color: "#545454"
divider-color: "rgba(255, 255, 255, 0.12)"
paper-card-background-color: "#4e4e4e"
paper-grey-200: "#191919"
paper-item-icon-color: "#d3d3d3"
paper-listbox-background-color: "#202020"
paper-listbox-color: "#FFFFFF"
primary-background-color: "#303030"
primary-text-color: "#cfcfcf"
secondary-background-color: "#2b2b2b"
secondary-text-color: "#04a7bc"
# Colors based on variables, see above
label-badge-background-color: "var(--secondary-background-color)"
label-badge-text-color: "var(--text-primary-color)"
paper-card-header-color: "#var(--paper-item-icon-color)"
paper-grey-50: "var(--primary-text-color)"
paper-item-icon-active-color: "var(--primary-color)"
paper-item-icon__color: "var(--primary-text-color)"
paper-slider-active-color: "var(--primary-color)"
paper-slider-knob-color: "var(--primary-color)"
paper-slider-knob-start-color: "var(--primary-color)"
paper-slider-pin-color: "var(--primary-color)"
paper-slider-secondary-color: "var(--light-primary-color)"
paper-toggle-button-checked-ink-color: "var(--dark-primary-color)"
paper-toggle-button-checked-button-color: "var(--primary-color)"
paper-toggle-button-checked-bar-color: "var(--light-primary-color)"
paper-toggle-button-unchecked-bar-color: "var(--primary-text-color)"
carbongreen:
#Color List
make-light-carbon: "#93A5AB"
make-carbon: "#111924"
make-grey: "#DEE5E7"
make-green: "#C2DD08"
make-blue: "#00A3DA"
make-red: "#EC5D57"
make-yellow: "#FCF933"
make-orange: "#E49D64"
make-white: "#FFFFFF"
make-disable: "#bdbdbd"
light-primary-color: "var(make-light-carbon)"
background-color: "var(--make-grey)"
dark-primary-color: "var(-make-carbon)"
#Main Colors
primary-background-color: "var(--make-light-carbon)" #Background
color settings
secondary-background-color: "var(--make-grey)" #Background tabs
primary-text-color: "var(--make-carbon)" #Primary Text
secondary-text-color: "var(--make-green)" #sub text etc.
disabled-text-color: "var(--make-disable)" #Deactivated Text
divider-color: "var(--make-green)" #Divider Linies
primary-color: "var(--make-carbon)" #Head of HA
#Labels
label-badge-red: "var(--make-red)"
label-badge-blue: "var(--make-blue)"
label-badge-green: "var(--make-green)"
label-badge-yellow: "var(--make-yellow)"
label-badge-grey: "var(--make-grey)"
label-badge-text-color: "var(--make-carbon)" #Text color sensors
label-badge-background-color: "var(--make-grey)" #Background sensors
#Google Loading

```

```

google-red-500: "var(--make-red)"
google-blue-500: "var(--make-blue)"
google-green-500: "var(--make-green)"
google-yellow-500: "var(--make-yellow)"
#Frontend
paper-green-400: "var(--make-green)"
paper-blue-400: "var(--make-blue)"
paper-orange-400: "var(--make-orange)"
paper-grey-50: "var(--make-grey)"
paper-grey-200: "var(--make-green)"
paper-item-icon-color: "var(--make-grey)" #Color Icon off
paper-item-icon-active-color: "var(--make-green)" #Color Icon on
paper-button-ink-color: "var(--make-carbon)" #Color Button selected
paper-button-color: "var(--make-red)"
paper-toggle-button-checked-ink-color: "var(--make-green)"
paper-toggle-button-checked-button-color: "var(--make-green)"
paper-toggle-button-checked-bar-color: "var(--make-green)"
paper-toggle-button-unchecked-button-color: "var(--make-grey)"
paper-toggle-button-unchecked-bar-color: "var(--make-grey)"
paper-toggle-button-unchecked-ink-color: "var(--make-grey)"
paper-slider-knob-color: "var(--make-carbon)" #Slider dot on
paper-slider-knob-start-color: "var(--make-carbon)" #Slider dot off
paper-slider-pin-color: "var(--make-red)" #
paper-slider-active-color: "var(--make-carbon)" #Slider line active
paper-slider-secondary-color: "var(--make-red)" #
paper-slider-container-color: "var(--make-light-carbon)" #Slider line
inactive
paper-listbox-color: "var(--make-carbon)" #Text Dorpdwn box
paper-listbox-background-color: "var(--make-grey)" #Background of
Listen
paper-card-background-color: "var(--make-light-carbon)" #Background
from cards in tabs
paper-card-header-color: "var(--make-carbon)" #Headline Color of tabs
paper-dialog-background-color: "var(--make-grey)" #Color Dialog Box
paper-item-selected_-_background-color: "var(--make-green)" #Selected
Option in Dropdown
paper-tabs-selection-bar-color: "var(--make-red)" #
table-row-background-color: "var(--make-grey)" #Background color
table
table-row-alternative-background-color: "var(--make-gery)" #second
color table
#Font
primary-font-family: "Gill Sans"
paper-font-common-base__font-family: "var(--primary-font-family)"
paper-font-common-code__font-family: "var(--primary-font-family)"
paper-font-body1__font-family: "var(--primary-font-family)"
paper-font-subhead__font-family: "var(--primary-font-family)"
paper-font-headline__font-family: "var(--primary-font-family)"
paper-font-caption__font-family: "var(--primary-font-family)"
paper-font-title__font-family: "var(--primary-font-family)"
PmxMononight:
# MyVariables
base-hue: '220' #Controls the base (and accent) color hue (0-360) |
0=Red 60=Yellow 120=Green 180=Cyan 240=Blue 300=Magenta 360=Red
base-sat: '16%' #Controls the saturation of the theme (0%-100%) |
0%=Grey 100%=Full Saturation
# MyVar
huesat: 'var(--base-hue), var(--base-sat),'
# Primary Color
primary-color: 'hsl(var(--huesat) 20%)'
# Backgrounds

```

```

primary-background-color: 'var(--primary-color)'
secondary-background-color: 'hsl(var(--huesat) 16%)'
paper-listbox-background-color: 'var(--primary-color)'
paper-card-background-color: 'hsl(var(--huesat) 12%)'
paper-dialog-background-color: 'var(--paper-card-background-color)'
table-row-background-color: 'hsl(var(--huesat) 12%)'
table-row-alternative-background-color: 'hsl(var(--huesat) 10%)'
# Divider
divider-color: 'hsla(0, 0%, 0%, 0)'
dark-divider-opacity: '0'
light-divider-opacity: '0'
# Text colors
primary-text-color: 'hsl(var(--huesat) 60%)'
text-primary-color: 'hsl(var(--huesat) 60%)'
secondary-text-color: 'hsl(var(--huesat) 60%)'
disabled-text-color: 'hsl(var(--huesat) 70%)'
sidebar-text__color: 'hsl(var(--huesat) 90%)'
sidebar-text-color: 'hsl(var(--huesat) 90%)'
paper-card-header-color: 'hsl(var(--base-hue), 90%, 50%)'
# Text Adjustments
paper-font-headline__letter-spacing: '-0.5px'
paper-font-headline__font-weight: '500'
paper-font-body1__font-weight: '500'
# Nav Menu
paper-listbox-color: 'hsl(var(--huesat) 50%)'
paper-grey-50: 'hsl(var(--huesat) 50%)'
paper-grey-200: 'hsla(var(--huesat) 26%)'
# Paper card
paper-item-icon-color: 'hsl(var(--huesat) 30%)'
paper-item-icon-active-color: 'var(--paper-item-icon-color)'
paper-item-icon__color: 'var(--paper-item-icon-color)'
paper-item-selected__background-color: 'hsla(0, 0%, 0%, 0.2)'
paper-tabs-selection-bar-color: 'hsla(0, 0%, 0%, 0.2)'
# Labels
label-badge-red: 'hsla(0, 0%, 0%, 0)'
label-badge-border-color: 'var(--label-badge-red)'
label-badge-background-color: 'var(--paper-card-background-color)'
label-badge-text-color: 'var(--primary-text-color)'
# Shadows
shadow-elevation-2dp__box-shadow: 'inset 0px 0px 0px 4px hsl(var(--
huesat) 18%)'
shadow-elevation-16dp__box-shadow: 'inset 0px 0px 0px 4px hsl(var(--
huesat) 28%)'
# Switches
paper-toggle-button-checked-button-color: 'hsl(var(--base-hue), 90%,
50%)'
paper-toggle-button-checked-bar-color: 'hsl(var(--huesat) 25%)'
paper-toggle-button-unchecked-button-color: 'hsl(var(--huesat) 25%)'
paper-toggle-button-unchecked-bar-color: 'hsl(var(--huesat) 5%)'
# Sliders
paper-slider-knob-color: 'hsl(var(--base-hue), 90%, 50%)'
paper-slider-knob-start-color: 'hsl(var(--base-hue), 80%, 25%)'
paper-slider-pin-color: 'hsl(var(--base-hue), 90%, 50%)'
paper-slider-active-color: 'hsl(var(--base-hue), 90%, 50%)'
paper-slider-container-color: 'hsl(var(--huesat) 28%)'
paper-slider-secondary-color: 'hsl(var(--huesat) 90%)'
paper-slider-disabled-active-color: 'hsl(var(--base-hue), 80%, 25%)'
paper-slider-disabled-secondary-color: 'hsl(var(--base-hue), 80%,
25%)'
paper-dialog-color: 'hsl(var(--base-hue), 20%, 80%)'
PmxMonolight:

```

```

# MyVar
huesat: '0, 0%, '
# Primary Color
primary-color: 'hsl(var(--huesat) 80%)'
dark-primary-color: 'hsl(var(--huesat) 20%)'
light-primary-color: 'hsl(var(--huesat) 90%)'
accent-color: 'hsl(var(--huesat) 30%)'
# Backgrounds
primary-background-color: 'var(--primary-color)'
secondary-background-color: 'hsl(var(--huesat) 86%)'
paper-listbox-background-color: 'var(--primary-color)'
paper-card-background-color: 'hsl(var(--huesat) 90%)'
paper-dialog-background-color: 'var(--paper-card-background-color)'
table-row-background-color: 'hsl(var(--huesat) 88%)'
table-row-alternative-background-color: 'hsl(var(--huesat) 90%)'
# Divider
divider-color: 'hsla(0, 0%, 0%, 0)'
dark-divider-opacity: '0'
light-divider-opacity: '0'
dark-secondary-opacity: '0.4'
# Text colors
primary-text-color: 'hsl(var(--huesat) 46%)'
text-primary-color: 'hsl(var(--huesat) 46%)'
secondary-text-color: 'hsl(var(--huesat) 56%)'
disabled-text-color: 'hsl(var(--huesat) 30%)'
sidebar-text_-_color: 'hsl(var(--huesat) 5%)'
sidebar-text-color: 'hsl(var(--huesat) 5%)'
paper-card-header-color: 'hsl(hsl(var(--huesat) 50%)'
paper-button-ink-color: 'hsl(hsl(var(--huesat) 50%)'
# Text Adjustments
paper-font-headline_-_letter-spacing: '-0.5px'
paper-font-headline_-_font-weight: '500'
paper-font-body1_-_font-weight: '500'
# Nav Menu
paper-listbox-color: 'hsl(var(--huesat) 50%)'
paper-grey-50: 'hsl(var(--huesat) 40%)'
paper-grey-200: 'hsl(var(--huesat) 72%)'
# Paper card
paper-item-icon-color: 'hsl(var(--huesat) 70%)'
paper-item-icon-active-color: 'var(--paper-item-icon-color)'
paper-item-icon_-_color: 'hsl(var(--huesat) 70%)'
paper-item-selected_-_background-color: 'hsla(0, 0%, 100%, 0.2)'
paper-item-selected_-_color: 'hsl(var(--huesat) 20%)'
paper-tabs-selection-bar-color: 'hsl(var(--huesat) 70%)'
paper-tab-ink: 'hsl(var(--huesat) 70%)'
paper-input-container-color: 'hsl(var(--huesat) 66%)'
# Labels
label-badge-red: 'hsl(var(--huesat) 80%)'
label-badge-border-color: 'var(--label-badge-red)'
label-badge-background-color: 'hsl(var(--huesat) 82%)'
label-badge-text-color: 'var(--primary-text-color)'
# Shadows
shadow-elevation-2dp_-_box-shadow: 'inset 0px 0px 0px 3px hsla(var(--huesat) 91%, 0.2)'
shadow-elevation-16dp_-_box-shadow: 'inset 0px 0px 0px 3px hsla(var(--huesat) 91%, 0.2)'
paper-input-container-shared-input-style_-_background: '#000'
# Switches
paper-toggle-button-checked-button-color: 'hsl(var(--huesat) 50%)'
paper-toggle-button-checked-bar-color: 'hsl(var(--huesat) 66%)'
paper-toggle-button-unchecked-button-color: 'hsl(var(--huesat) 85%)'

```

```

    paper-toggle-button-unchecked-bar-color: 'hsl(var(--huesat) 80%)'
    # Sliders
    paper-slider-knob-color: 'hsl(var(--huesat) 45%)'
    paper-slider-knob-start-color: 'hsl(var(--huesat) 60%)'
    paper-slider-pin-color: 'hsl(var(--huesat) 60%)'
    paper-slider-active-color: 'hsl(var(--huesat) 60%)'
    paper-slider-container-color: 'hsl(var(--huesat) 76%)'
    paper-slider-secondary-color: 'hsl(var(--huesat) 10%)'
    paper-slider-disabled-active-color: 'hsl(var(--huesat) 75%)'
    paper-slider-disabled-secondary-color: 'hsl(var(--huesat) 75%)'
    #Toast
    paper-toast-color: 'hsl(var(--huesat) 45%)'
    paper-toast-background-color: 'hsl(var(--huesat) 90%)'

config:

http:
  api_password: *****

# Checks for available updates
# Note: This component will send some information about your system to
# the developers to assist with development of Home Assistant.
# For more information, please see:
# https://home-assistant.io/blog/2016/10/25/explaining-the-updater/
updater:
  # Optional, allows Home Assistant developers to focus on popular
  # components.
  # include_used_components: true

# Discover some devices automatically
discovery:

# Allows you to issue voice commands from the frontend in enabled browsers
conversation:

# Enables support for tracking state changes over time
history:

# View all events in a logbook
logbook:

# Enables a map showing the location of tracked devices
map:

# Track the sun
sun:

#####
##notify
#####
notify:
  - platform: ios
    name: ios
    target: '709af45f-31d-8cc7-b8a8c086e045'

#####
##MQTT
#####
mqtt:

```

```

broker: ***.***.*.*
username: ****
password: ****

#####
##IFTTT
#####
ifttt:
    key: iAh15wymU0xeY-RZzePhe

#####
##input
#####
input_boolean:
    nightlight:
        name: nightlight
        initial: on
        icon: mdi:sleep
    lightturnon:
        name: lightturnon
        initial: off
    findphone:
        name: findphone
        initial: off
    alarmclock_enabled:
        name: Enabled
        initial: on
        icon: mdi:toggle-switch

input_number:
    alarmclock_hour:
        name: Hour
        icon: mdi:timer
        initial: 7
        min: 0
        max: 23
        step: 1
    alarmclock_minute:
        name: Minute
        icon: mdi:timer
        initial: 30
        min: 0
        max: 55
        step: 5

#####
##Zwave
#####
zwave:
    usb_path: /dev/ttyACM0

ios:

#####
##IP cam
#####
camera:

```

```

- platform: mjpeg
  mjpeg_url: http://192.168.0.185:8080/video

- platform: generic
  name: ISS
  still_image_url:
https://maps.googleapis.com/maps/api/staticmap?center={{
states.binary_sensor.iss.attributes.lat }},{{
states.binary_sensor.iss.attributes.long
}}&zoom=5&size=500x500&maptype=roadmap&markers=color:blue%7Clabel:P%7C{{
states.binary_sensor.iss.attributes.lat }},{{
states.binary_sensor.iss.attributes.long }}
  limit_refetch_to_url_change: true

#####
##ISS
#####
binary_sensor:
- platform: iss
  show_on_map: true

#####
##sensors
#####
sensor:
- platform: yr
  name: Weather
  monitored_conditions:
    - temperature
    - symbol
    - windSpeed
    - pressure
    - windDirection
    - humidity
    - fog
    - cloudiness
    - dewpointTemperature

- platform: bitcoin
  display_options:
    - exchangerate
    - market_price_usd

- platform: moon

- platform: mqtt
  state_topic: "bruh/sensornode1"
  name: "motion"
  value_template: '{{ value_json.motion }}'
  icon: mdi:walk

- platform: mqtt
  state_topic: "bruh/sensornode1"
  name: "Light"
  value_template: '{{ value_json.ldr }}'
  icon: mdi:vanish

- platform: mqtt
  state_topic: "bruh/sensornode1"

```

```

    name: "Humidity"
    unit_of_measurement: "%"
    value_template: '{{ value_json.humidity | round(1) }}'

- platform: mqtt
  state_topic: "bruh/sensornodel"
  name: "Temperature"
  unit_of_measurement: "°F"
  value_template: '{{ value_json.temperature | round(1) }}'

- platform: template
  sensors:
    alarmclock_time_template:
      friendly_name: 'Time'
      icon_template: "{{ 'mdi:alarm' if
is_state('input_boolean.alarmclock_enabled', 'on') else 'mdi:close-circle'
}}"
      value_template: '{{ "%0.02d:%0.02d" |
format(states("input_number.alarmclock_hour") | int,
states("input_number.alarmclock_minute") | int) }}'

#####
##text to speech
#####
tts:
  - platform: google

#####
##switches
#####
switch:

- platform: mqtt
  name: 'Office Switch'
  state_topic: 'office/switch1/status'
  command_topic: 'office/switch1/set'
  retain: true
  optimistic: false

- platform: command_line
  switches:
    aeotec_zstick_disco_light:
      friendly_name: 'Aeotec Z-Stick Disco Light'
      command_on: 'echo -e -n "\x01\x08\x00\xF2\x51\x01\x01\x05\x01\x50"
> /dev/serial/by-id/usb-0658_0200-if00'
      command_off: 'echo -e -n "\x01\x08\x00\xF2\x51\x01\x00\x05\x01\x51"
> /dev/serial/by-id/usb-0658_0200-if00'

switch 1:
  - platform: mqtt
    name: "MQTT Switch 1"
    state_topic: "/house/switchConfirm1/"
    command_topic: "/house/switch1/"
    payload_on: "1"
    payload_off: "0"
    qos: 0
    retain: true

switch 2:

```

```

- platform: mqtt
  name: "MQTT Switch 2"
  state_topic: "/house/switchConfirm2/"
  command_topic: "/house/switch2/"
  payload_on: "1"
  payload_off: "0"
  qos: 0
  retain: true

switch 3:
- platform: mqtt
  name: "MQTT Switch 3"
  state_topic: "/house/switchConfirm3/"
  command_topic: "/house/switch3/"
  payload_on: "1"
  payload_off: "0"
  qos: 0
  retain: true

switch 4:
- platform: mqtt
  name: "MQTT Switch 4"
  state_topic: "/house/switchConfirm4/"
  command_topic: "/house/switchwatch/"
  payload_on: "1"
  payload_off: "0"
  qos: 0
  retain: true

#####
##media
#####
media_player:
- platform: spotify
  client_id: 17ab51d311f342c1aeda76bc9ee650b9
  client_secret: a7de63354b4e4fb98b328f76aa01ede8

- platform: sonos
  interface_addr: 192.168.0.109

#####
##include
#####
group: !include groups.yaml
automation: !include automations.yaml
script: !include scripts.yaml

#####
##light
#####
light:
- platform: mqtt_json
  name: bed light
  state_topic: "home/rgb1"
  command_topic: "home/rgb1/set"
  brightness: true
  rgb: true
  transition: 1000

```

```

- platform: yeelight
  devices:
    192.168.0.252:
      name: yeeeeeeeeee
      transition: 1000
      use_music_mode: False #(defaults to False)
      save_on_change: True

- platform: mqtt
  name: 'Office light'
  state_topic: 'office/light'
  command_topic: 'office/light/switch'
  brightness_state_topic: 'office/light/brightness'
  brightness_command_topic: 'office/light/brightness/set'
  optimistic: false

#####
##nest
#####
nest:
  client_id: fa7192a7-db43-40d0-9ea4-be03b1a6b15f
  client_secret: i4wrr4eYImrSGuWwPm9MF9kxj

#####
##bluetooth tracker
#####
device_tracker:
  - platform: bluetooth_tracker

#####
##weather
#####
weather:
  - platform: buienradar
    name: 'Witgoor'
    latitude: 51.24285187577013
    longitude: 5.139476833869833
    forecast: True

#####
##automations
#####
automation:
  - alias: motion alarm
    trigger:
      - platform: state
        entity_id: sensor.Motion
        to: 'ON'
    condition:
      - condition: state
        entity_id: device_tracker.iphone_van_lian
        state: 'not_home'
    action:
      - service: ifttt.trigger
        data: {"event": "motion"}

```

```

- alias: LostPhone
  trigger:
  - platform: state
    entity_id: input_boolean.findphone
    to: 'on'
  action:
    - service: ifttt.trigger
      data: {"event":"phone"}

- alias: 'dimmer on'
  trigger:
    - platform: state
      entity_id: switch.office_switch
      to: 'on'
  action:
    - service: light.toggle
      entity_id: light.office_light

- alias: 'dimmer on2'
  trigger:
    - platform: state
      entity_id: switch.office_switch
      to: 'off'
  action:
    - service: light.toggle
      entity_id: light.office_light
#
#####
##automation nightlight###
#####
- alias: 'motion yee'
  trigger:
    - platform: state
      entity_id: sensor.Motion
      to: 'ON'
  condition:
    - condition: numeric_state
      entity_id: sensor.light
      below: 70
    - condition: state
      entity_id: input_boolean.Nightlight
      state: 'on'
    - condition: state
      entity_id: light.Bed_light
      state: 'off'
    - condition: state
      entity_id: light.Yeeeeeeeeee
      state: 'off'
  action:
    - service: light.turn_on
      entity_id: light.Yeeeeeeeeee
      data:
        brightness: 5
        rgb_color: [255, 155, 0]
        transition: 5
    - service: input_boolean.turn_on
      data:
        entity_id: input_boolean.lightturnon

- alias: 'motion bed'
  trigger:

```

```

    - platform: state
      entity_id: sensor.Motion
      to: 'ON'
condition:
- condition: numeric_state
  entity_id: sensor.light
  below: 70
- condition: state
  entity_id: input_boolean.Nightlight
  state: 'on'
- condition: state
  entity_id: light.Bed_light
  state: 'off'
- condition: state
  entity_id: light.Yeeeeeeeeee
  state: 'off'
action:
  service: light.turn_on
  entity_id: light.Bed_light
  data:
    brightness: 30
    rgb_color: [255, 100, 0]
    transition: 5

- alias: 'motionuityee'
  trigger:
    - platform: state
      entity_id: sensor.Motion
      to: 'OFF'
  condition:
    - condition: state
      entity_id: input_boolean.Nightlight
      state: 'on'
    - condition: state
      entity_id: input_boolean.lightturnon
      state: 'on'
  action:
    service: script.turn_on
    entity_id: script.motionuityeescript

- alias: 'motionuitbed'
  trigger:
    - platform: state
      entity_id: sensor.Motion
      to: 'OFF'
  condition:
    - condition: state
      entity_id: input_boolean.Nightlight
      state: 'on'
    - condition: state
      entity_id: input_boolean.lightturnon
      state: 'on'
  action:
    service: script.turn_on
    entity_id: script.motionuitbedscript
##### #
##automation nightlight ###
##### #
##### #

- alias: Alarm Clock
  trigger:

```

```

- platform: time
  minutes: '/1'
  seconds: 00
condition:
- condition: and
  conditions:
- condition: template
  value_template: '{{ now().time().strftime("%H:%M") ==
states.sensor.alarmclock_time_template.state }}'
- condition: state
  entity_id: input_boolean.alarmclock_enabled
  state: 'on'
action:
  service: light.turn_on
  entity_id: light.nano_level
  data:
    transition: 900

#####
##scripts
#####
script:
  motionuitbedscript:
    alias: motionuit bed script
    sequence:
      - delay:
          # supports seconds, milliseconds, minutes, hours
          seconds: 60
      - alias: Living room lights on
        service: light.turn_off
        data:
          entity_id: light.Bed_light
      - alias: reset
        service: input_boolean.turn_off
        data:
          entity_id: input_boolean.lightturnon

  motionuityeescript:
    alias: motionuit yee script
    sequence:
      - delay:
          # supports seconds, milliseconds, minutes, hours
          seconds: 60
      - alias: Living room lights on 2
        service: light.turn_off
        data:
          entity_id: light.Yeeeeeeeeee

#####
##scene
#####
scene:
- name: Movie
  entities:
    light.office_light:
      state: on
      brightness: 30
    light.Bed_light:
      state: on

```

```

    rgb_color: [255, 100, 0]
    brightness: 255
light.Yeeeeeeeeee:
    state: on
    rgb_color: [255, 150, 0]
    brightness: 100
light.nano_level: off

#####
##zone
#####
zone:
- name: School
  latitude: 51.176522
  longitude: 5.124985
  radius: 150
  icon: mdi:school
- name: Joyce
  latitude: 51.171448
  longitude: 5.152784
  radius: 20
  icon: mdi:heart

#####
##group
#####
group:
  default_view:
    view: yes
    icon: mdi:home
    entities:
      - group.light
      - group.sensors
      - group.alarmclock
      - group.thuis
      - group.miscellaneous
      - group.media
      - group.weather
      - group.automation

light:
  name: light
  entities:
    - light.nano_level
    - light.Yeeeeeeeeee
    - light.office_light
    - light.Bed_light
    - climate.downstairs
    - input_select.set_current_theme

sensors:
  name: sensors
  entities:
    - sensor.Motion
    - sensor.light
    - sensor.humidity
    - sensor.temperature
    - device_tracker.iphone_van_lian

```

```

alarmclock:
  name: Alarm Clock
  entities:
    - sensor.alarmclock_time_template
    - input_boolean.alarmclock_enabled
    - input_number.alarmclock_hour
    - input_number.alarmclock_minute

miscellaneous:
  name: miscellaneous
  entities:
    - sensor.exchange_rate_1_btc
    - binary_sensor.iss
    - camera.iss

media:
  name: media
  entities:
    - remote.afstandsbediening
    - media_player.aawasoundtouch_speaker
    - media_player.spotify
    - media_player.apple_tv
    - media_player.huiskamer
    - media_player.lians_bedroom_speaker_3

weather:
  name: weather
  entities:
    - sun.sun
    - sensor.weather_cloudiness
    - weather.witgoor

automation:
  name: automation
  entities:
    - automation.opstaan
    - switch.aeotec_zstick_disco_light
    - sensor.iphone_van_lian_battery_level
    - camera.mjpeg_camera
    - input_boolean.Nightlight
    - input_boolean.findphone

all:
  view: yes
  icon: mdi:death-star
  entities:
    - group.all
    - group.all_automations
    - group.all_devices
    - group.all_lights
    - group.all_remotes
    - group.all_scripts
    - group.all_switches

```

6.2 Arduino code dimmer

```
const byte interruptPin = 15;
int val = 0;
int encoder0PinA = 13;
int encoder0PinB = 12;
int n = LOW;
int encoderposb = LOW;
int encoder0PinALast = LOW;

const PROGMEM uint8_t BUTTON_PIN = 14;

#include <OneButton.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define MQTT_VERSION MQTT_VERSION_3_1_1

OneButton button(BUTTON_PIN, false);

const char*          WIFI_SSID          = "Eminent";
const char*          WIFI_PASSWORD      = "Tisp2016";

// MQTT: ID, server IP, port, username and password
const PROGMEM char*   MQTT_CLIENT_ID    = "office_light";
const PROGMEM char*   MQTT_SERVER_IP    = "192.168.0.50";
const PROGMEM uint16_t MQTT_SERVER_PORT = 1883;
const PROGMEM char*   MQTT_USER         = "lian";
const PROGMEM char*   MQTT_PASSWORD     = "aarts";

// MQTT: topics
// state
const PROGMEM char*   MQTT_LIGHT_STATE_TOPIC          =
"office/light";
const PROGMEM char*   MQTT_LIGHT_COMMAND_TOPIC        =
"office/light/switch";

// brightness
const PROGMEM char*   MQTT_LIGHT_BRIGHTNESS_STATE_TOPIC =
"office/light/brightness";
const PROGMEM char*   MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC =
"office/light/brightness/set";

// payloads by default (on/off)
const PROGMEM char*   LIGHT_ON          = "ON";
const PROGMEM char*   LIGHT_OFF         = "OFF";

// variables used to store the state and the brightness of the light
boolean  m_light_state      = false;
uint8_t  m_light_brightness = 255;

// buffer used to send/receive data with MQTT
const uint8_t MSG_BUFFER_SIZE = 20;
char m_msg_buffer[MSG_BUFFER_SIZE];

const PROGMEM char* MQTT_SWITCH_STATUS_TOPIC = "office/switch1/status";
const PROGMEM char* MQTT_SWITCH_COMMAND_TOPIC = "office/switch1/set";
```

```

// default payload
const PROGMEM char* SWITCH_ON = "ON";
const PROGMEM char* SWITCH_OFF = "OFF";
boolean m_switch_state = false;

WiFiClient wifiClient;
PubSubClient client(wifiClient);

void setup() {
  ESP.wdtDisable();
  ESP.wdtEnable(8000);
  // init the serial
  Serial.begin(115200);

  // init the led
  pinMode(4, OUTPUT);
  analogWriteRange(255);
  setLightState();

  button.attachClick(click);

  // init the WiFi connection
  Serial.println();
  Serial.println();
  Serial.print("INFO: Connecting to ");
  WiFi.mode(WIFI_STA);
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("INFO: WiFi connected");
  Serial.print("INFO: IP address: ");
  Serial.println(WiFi.localIP());

  // init the MQTT connection
  client.setServer(MQTT_SERVER_IP, MQTT_SERVER_PORT);
  client.setCallback(callback);

  pinMode(interruptPin, INPUT);
  attachInterrupt(digitalPinToInterrupt(interruptPin), nuldoorgang,
  RISING);

  pinMode(encoder0PinA, INPUT);
  pinMode(encoder0PinB, INPUT);
}

void loop() {
  button.tick();

  if (m_light_state == true) {
    attachInterrupt(digitalPinToInterrupt(interruptPin), nuldoorgang,
    RISING);
  }
  if (m_light_state == false) {

```

```

    detachInterrupt(digitalPinToInterrupt(interruptPin));
}

if (!client.connected()) {
    reconnect();
}
client.loop();

n = digitalRead(encoder0PinA);
encoderposb = digitalRead(encoder0PinB);

if ((encoder0PinALast == LOW) && (n == HIGH)) {
    if (encoderposb == LOW) {
        m_light_brightness = m_light_brightness - 4;
    }

    else {
        m_light_brightness = m_light_brightness - 4;
    }
}
encoder0PinALast = n;
}

void click() {
    if (m_switch_state) {
        m_switch_state = false;
    } else {
        m_switch_state = true;
    }
    publishSwitchState();
}

void publishSwitchState() {
    if (m_switch_state) {
        client.publish(MQTT_SWITCH_STATUS_TOPIC, SWITCH_ON, true);
    } else {
        client.publish(MQTT_SWITCH_STATUS_TOPIC, SWITCH_OFF, true);
    }
}

void nuldoorgang() {
    val = map(m_light_brightness, 0, 255, 8000, 1);

    if (m_light_state == true) {
        delayMicroseconds(val);
        digitalWrite(4, HIGH);
        delayMicroseconds(250);
        digitalWrite(4, LOW);
    }

    if (m_light_state == false) {
        digitalWrite(4, LOW);
    }
}

void setLightState() {

```

```

    if (m_light_state) {
        Serial.print("INFO: Brightness: ");
        Serial.println(m_light_brightness);
    } else {
        Serial.println("INFO: Turn light off");
    }
}

// function called to publish the state of the led (on/off)
void publishLightState() {
    if (m_light_state) {
        client.publish(MQTT_LIGHT_STATE_TOPIC, LIGHT_ON, true);
    } else {
        client.publish(MQTT_LIGHT_STATE_TOPIC, LIGHT_OFF, true);
    }
}

// function called to publish the brightness of the led
void publishLightBrightness() {
    snprintf(m_msg_buffer, MSG_BUFFER_SIZE, "%d", m_light_brightness);
    client.publish(MQTT_LIGHT_BRIGHTNESS_STATE_TOPIC, m_msg_buffer, true);
}

// function called when a MQTT message arrived
void callback(char* p_topic, byte* p_payload, unsigned int p_length) {
    // concat the payload into a string
    String payload;
    for (uint8_t i = 0; i < p_length; i++) {
        payload.concat((char)p_payload[i]);
    }
    // handle message topic
    if (String(MQTT_LIGHT_COMMAND_TOPIC).equals(p_topic)) {
        // test if the payload is equal to "ON" or "OFF"
        if (payload.equals(String(LIGHT_ON))) {
            if (m_light_state != true) {
                m_light_state = true;
                setLightState();
                publishLightState();
            }
        } else if (payload.equals(String(LIGHT_OFF))) {
            if (m_light_state != false) {
                m_light_state = false;
                setLightState();
                publishLightState();
            }
        }
    }
    } else if (String(MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC).equals(p_topic)) {
        uint8_t brightness = payload.toInt();
        if (brightness < 0 || brightness > 255) {
            // do nothing...
            return;
        } else {
            m_light_brightness = brightness;
            setLightState();
            publishLightBrightness();
        }
    }

    if (String(MQTT_SWITCH_COMMAND_TOPIC).equals(p_topic)) {
        // test if the payload is equal to "ON" or "OFF"

```

```

    if (payload.equals(String(SWITCH_ON))) {
        if (m_switch_state != true) {
            m_switch_state = true;
            publishSwitchState();
        }
    } else if (payload.equals(String(SWITCH_OFF))) {
        if (m_switch_state != false) {
            m_switch_state = false;
            publishSwitchState();
        }
    }
}

}

}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.println("INFO: Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(MQTT_CLIENT_ID, MQTT_USER, MQTT_PASSWORD)) {
            Serial.println("\nINFO: connected");

            // Once connected, publish an announcement...
            // publish the initial values
            publishLightState();
            publishLightBrightness();

            // ... and resubscribe
            client.subscribe(MQTT_LIGHT_COMMAND_TOPIC);
            client.subscribe(MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC);
        } else {
            Serial.print("ERROR: failed, rc=");
            Serial.print(client.state());
            Serial.println("DEBUG: try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
}

```

6.3 Arduino code rgb controller

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define MQTT_VERSION MQTT_VERSION_3_1_1

// Wifi: SSID and password
const char* WIFI_SSID = "Eminent";
const char* WIFI_PASSWORD = "Tisp2016";

// MQTT: ID, server IP, port, username and password
const PROGMEM char* MQTT_CLIENT_ID = "office_rgb_light";
const PROGMEM char* MQTT_SERVER_IP = "192.168.0.50";
const PROGMEM uint16_t MQTT_SERVER_PORT = 1883;
const PROGMEM char* MQTT_USER = "lian";
const PROGMEM char* MQTT_PASSWORD = "aarts";

// MQTT: topics
// state
const PROGMEM char* MQTT_LIGHT_STATE_TOPIC = "office/rgb1/light/status";
const PROGMEM char* MQTT_LIGHT_COMMAND_TOPIC = "office/rgb1/light/switch";

// brightness
const PROGMEM char* MQTT_LIGHT_BRIGHTNESS_STATE_TOPIC =
"office/rgb1/brightness/status";
const PROGMEM char* MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC =
"office/rgb1/brightness/set";

// colors (rgb)
const PROGMEM char* MQTT_LIGHT_RGB_STATE_TOPIC = "office/rgb1/rgb/status";
const PROGMEM char* MQTT_LIGHT_RGB_COMMAND_TOPIC = "office/rgb1/rgb/set";

// payloads by default (on/off)
const PROGMEM char* LIGHT_ON = "ON";
const PROGMEM char* LIGHT_OFF = "OFF";

// variables used to store the state, the brightness and the color of the
light
boolean m_rgb_state = false;
uint8_t m_rgb_brightness = 100;
uint8_t m_rgb_red = 255;
uint8_t m_rgb_green = 255;
uint8_t m_rgb_blue = 255;

// pins used for the rgb led (PWM)
const PROGMEM uint8_t RGB_LIGHT_RED_PIN = 0;
const PROGMEM uint8_t RGB_LIGHT_GREEN_PIN = 5;
const PROGMEM uint8_t RGB_LIGHT_BLUE_PIN = 4;

// buffer used to send/receive data with MQTT
const uint8_t MSG_BUFFER_SIZE = 20;
char m_msg_buffer[MSG_BUFFER_SIZE];

WiFiClient wifiClient;
PubSubClient client(wifiClient);

// function called to adapt the brightness and the color of the led
void setColor(uint8_t p_red, uint8_t p_green, uint8_t p_blue) {
    // convert the brightness in % (0-100%) into a digital value (0-255)
```

```

uint8_t brightness = map(m_rgb_brightness, 0, 100, 0, 255);

analogWrite(RGB_LIGHT_RED_PIN, map(p_red, 0, 255, 0, brightness));
analogWrite(RGB_LIGHT_GREEN_PIN, map(p_green, 0, 255, 0, brightness));
analogWrite(RGB_LIGHT_BLUE_PIN, map(p_blue, 0, 255, 0, brightness));
}

// function called to publish the state of the led (on/off)
void publishRGBState() {
    if (m_rgb_state) {
        client.publish(MQTT_LIGHT_STATE_TOPIC, LIGHT_ON, true);
    } else {
        client.publish(MQTT_LIGHT_STATE_TOPIC, LIGHT_OFF, true);
    }
}

// function called to publish the brightness of the led (0-100)
void publishRGBBrightness() {
    snprintf(m_msg_buffer, MSG_BUFFER_SIZE, "%d", m_rgb_brightness);
    client.publish(MQTT_LIGHT_BRIGHTNESS_STATE_TOPIC, m_msg_buffer, true);
}

// function called to publish the colors of the led (xx(x),xx(x),xx(x))
void publishRGBColor() {
    snprintf(m_msg_buffer, MSG_BUFFER_SIZE, "%d,%d,%d", m_rgb_red,
m_rgb_green, m_rgb_blue);
    client.publish(MQTT_LIGHT_RGB_STATE_TOPIC, m_msg_buffer, true);
}

// function called when a MQTT message arrived
void callback(char* p_topic, byte* p_payload, unsigned int p_length) {
    // concat the payload into a string
    String payload;
    for (uint8_t i = 0; i < p_length; i++) {
        payload.concat((char)p_payload[i]);
    }
    // handle message topic
    if (String(MQTT_LIGHT_COMMAND_TOPIC).equals(p_topic)) {
        // test if the payload is equal to "ON" or "OFF"
        if (payload.equals(String(LIGHT_ON))) {
            if (m_rgb_state != true) {
                m_rgb_state = true;
                setColor(m_rgb_red, m_rgb_green, m_rgb_blue);
                publishRGBState();
            }
        } else if (payload.equals(String(LIGHT_OFF))) {
            if (m_rgb_state != false) {
                m_rgb_state = false;
                setColor(0, 0, 0);
                publishRGBState();
            }
        }
    }
    else if (String(MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC).equals(p_topic)) {
        uint8_t brightness = payload.toInt();
        if (brightness < 0 || brightness > 100) {
            // do nothing...
            return;
        } else {
            m_rgb_brightness = brightness;
            setColor(m_rgb_red, m_rgb_green, m_rgb_blue);
            publishRGBBrightness();
        }
    }
}

```

```

    }
} else if (String(MQTT_LIGHT_RGB_COMMAND_TOPIC).equals(p_topic)) {
    // get the position of the first and second commas
    uint8_t firstIndex = payload.indexOf(',');
    uint8_t lastIndex = payload.lastIndexOf(',');

    uint8_t rgb_red = payload.substring(0, firstIndex).toInt();
    if (rgb_red < 0 || rgb_red > 255) {
        return;
    } else {
        m_rgb_red = rgb_red;
    }

    uint8_t rgb_green = payload.substring(firstIndex + 1,
lastIndex).toInt();
    if (rgb_green < 0 || rgb_green > 255) {
        return;
    } else {
        m_rgb_green = rgb_green;
    }

    uint8_t rgb_blue = payload.substring(lastIndex + 1).toInt();
    if (rgb_blue < 0 || rgb_blue > 255) {
        return;
    } else {
        m_rgb_blue = rgb_blue;
    }

    setColor(m_rgb_red, m_rgb_green, m_rgb_blue);
    publishRGBColor();
}
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("INFO: Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(MQTT_CLIENT_ID, MQTT_USER, MQTT_PASSWORD)) {
            Serial.println("INFO: connected");

            // Once connected, publish an announcement...
            // publish the initial values
            publishRGBState();
            publishRGBBrightness();
            publishRGBColor();

            // ... and resubscribe
            client.subscribe(MQTT_LIGHT_COMMAND_TOPIC);
            client.subscribe(MQTT_LIGHT_BRIGHTNESS_COMMAND_TOPIC);
            client.subscribe(MQTT_LIGHT_RGB_COMMAND_TOPIC);
        } else {
            Serial.print("ERROR: failed, rc=");
            Serial.print(client.state());
            Serial.println("DEBUG: try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
}

```

```

void setup() {
  // init the serial
  Serial.begin(115200);

  // init the RGB led
  pinMode(RGB_LIGHT_BLUE_PIN, OUTPUT);
  pinMode(RGB_LIGHT_RED_PIN, OUTPUT);
  pinMode(RGB_LIGHT_GREEN_PIN, OUTPUT);
  analogWriteRange(255);
  setColor(0, 0, 0);

  // init the WiFi connection
  Serial.println();
  Serial.println();
  Serial.print("INFO: Connecting to ");
  WiFi.mode(WIFI_STA);
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("INFO: WiFi connected");
  Serial.print("INFO: IP address: ");
  Serial.println(WiFi.localIP());

  // init the MQTT connection
  client.setServer(MQTT_SERVER_IP, MQTT_SERVER_PORT);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}

#include <ESP8266WiFi.h>
#include <DHT.h>
#include <PubSubClient.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ArduinoJson.h>

/***** WIFI and MQTT INFORMATION (CHANGE THESE FOR YOUR SETUP) *****/
#define wifi_ssid "Eminent" //type your WIFI information inside the quotes
#define wifi_password "Tisp2016"
#define mqtt_server "192.168.0.50"
#define mqtt_user "lian"

```

```

#define mqtt_password "aarts"
#define mqtt_port 1883

/***** MQTT TOPICS (change these topics as you
wish) *****/
#define light_state_topic "bruh/sensornode1"
#define light_set_topic "bruh/sensornode1/set"

const char* on_cmd = "ON";
const char* off_cmd = "OFF";

/***** FOR OTA *****/
#define SENSORNAME "sensornode1"
#define OTApassword "YouPassword" // change this to whatever password you
want to use when you upload OTA
int OTAport = 8266;

/***** PIN DEFINITIONS *****/
const int redPin = D1;
const int greenPin = D2;
const int bluePin = D3;
#define PIRPIN D5
#define DHTPIN D7
#define DHTTYPE DHT22
#define LDRPIN A0

/***** SENSOR DEFINITIONS *****/
float ldrValue;
int LDR;
float calcLDR;
float diffLDR = 25;

float diffTEMP = 0.2;
float tempValue;

float diffHUM = 1;
float humValue;

int pirValue;
int pirStatus;
String motionStatus;

char message_buff[100];

int calibrationTime = 0;

const int BUFFER_SIZE = 300;

#define MQTT_MAX_PACKET_SIZE 512

```

```

/***** GLOBALS for fade/flash
*****/
byte red = 255;
byte green = 255;
byte blue = 255;
byte brightness = 255;

byte realRed = 0;
byte realGreen = 0;
byte realBlue = 0;

bool stateOn = false;

bool startFade = false;
unsigned long lastLoop = 0;
int transitionTime = 0;
bool inFade = false;
int loopCount = 0;
int stepR, stepG, stepB;
int redVal, grnVal, bluVal;

bool flash = false;
bool startFlash = false;
int flashLength = 0;
unsigned long flashStartTime = 0;
byte flashRed = red;
byte flashGreen = green;
byte flashBlue = blue;
byte flashBrightness = brightness;

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

/***** START
SETUP*****/
void setup() {

    Serial.begin(115200);

    pinMode(PIRPIN, INPUT);
    pinMode(DHTPIN, INPUT);
    pinMode(LDRPIN, INPUT);

    Serial.begin(115200);
    delay(10);

    ArduinoOTA.setPort(OTApport);

    ArduinoOTA.setHostname(SENSORNAME);

    ArduinoOTA.setPassword((const char *)OTApasword);

    Serial.print("calibrating sensor ");
    for (int i = 0; i < calibrationTime; i++) {
        Serial.print(".");
    }
}

```

```

    delay(1000);
}

Serial.println("Starting Node named " + String(SENSORNAME));

setup_wifi();

client.setServer(mqtt_server, mqtt_port);
client.setCallback(callback);

ArduinoOTA.onStart([]() {
    Serial.println("Starting");
});
ArduinoOTA.onEnd([]() {
    Serial.println("\nEnd");
});
ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
});
ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
});
ArduinoOTA.begin();
Serial.println("Ready");
Serial.print("IPess: ");
Serial.println(WiFi.localIP());
reconnect();
}

/***** START SETUP
WIFI*****/
void setup_wifi() {

    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);

    WiFi.mode(WIFI_STA);
    WiFi.begin(wifi_ssid, wifi_password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

```

```

/***** START
CALLBACK*****/
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");

    char message[length + 1];
    for (int i = 0; i < length; i++) {
        message[i] = (char)payload[i];
    }
    message[length] = '\0';
    Serial.println(message);

    if (!processJson(message)) {
        return;
    }

    if (stateOn) {
        // Update lights
        realRed = map(red, 0, 255, 0, brightness);
        realGreen = map(green, 0, 255, 0, brightness);
        realBlue = map(blue, 0, 255, 0, brightness);
    }
    else {
        realRed = 0;
        realGreen = 0;
        realBlue = 0;
    }

    startFade = true;
    inFade = false; // Kill the current fade

    sendState();
}

/***** START PROCESS
JSON*****/
bool processJson(char* message) {
    StaticJsonBuffer<BUFFER_SIZE> jsonBuffer;

    JsonObject& root = jsonBuffer.parseObject(message);

    if (!root.success()) {
        Serial.println("parseObject() failed");
        return false;
    }

    if (root.containsKey("state")) {
        if (strcmp(root["state"], on_cmd) == 0) {
            stateOn = true;
        }
        else if (strcmp(root["state"], off_cmd) == 0) {
            stateOn = false;
        }
    }
}

```

```

// If "flash" is included, treat RGB and brightness differently
if (root.containsKey("flash")) {
    flashLength = (int)root["flash"] * 1000;

    if (root.containsKey("brightness")) {
        flashBrightness = root["brightness"];
    }
    else {
        flashBrightness = brightness;
    }

    if (root.containsKey("color")) {
        flashRed = root["color"]["r"];
        flashGreen = root["color"]["g"];
        flashBlue = root["color"]["b"];
    }
    else {
        flashRed = red;
        flashGreen = green;
        flashBlue = blue;
    }

    flashRed = map(flashRed, 0, 255, 0, flashBrightness);
    flashGreen = map(flashGreen, 0, 255, 0, flashBrightness);
    flashBlue = map(flashBlue, 0, 255, 0, flashBrightness);

    flash = true;
    startFlash = true;
}
else { // Not flashing
    flash = false;

    if (root.containsKey("color")) {
        red = root["color"]["r"];
        green = root["color"]["g"];
        blue = root["color"]["b"];
    }

    if (root.containsKey("brightness")) {
        brightness = root["brightness"];
    }

    if (root.containsKey("transition")) {
        transitionTime = root["transition"];
    }
    else {
        transitionTime = 0;
    }
}

return true;
}

/***** START SEND
STATE*****/
void sendState() {
    StaticJsonBuffer<BUFFER_SIZE> jsonBuffer;

```

```

JsonObject& root = jsonBuffer.createObject();

root["state"] = (stateOn) ? on_cmd : off_cmd;
JsonObject& color = root.createNestedObject("color");
color["r"] = red;
color["g"] = green;
color["b"] = blue;


root["brightness"] = brightness;
root["humidity"] = (String)humValue;
root["motion"] = (String)motionStatus;
root["ldr"] = (String)LDR;
root["temperature"] = (String)tempValue;
root["heatIndex"] = (String)calculateHeatIndex(humValue, tempValue);


char buffer[root.measureLength() + 1];
root.printTo(buffer, sizeof(buffer));

Serial.println(buffer);
client.publish(light_state_topic, buffer, true);
}

/*
 * Calculate Heat Index value AKA "Real Feel"
 * NOAA heat index calculations taken from
 * http://www.wpc.ncep.noaa.gov/html/heatindex\_equation.shtml
 */
float calculateHeatIndex(float humidity, float temp) {
    float heatIndex= 0;
    if (temp >= 80) {
        heatIndex = -42.379 + 2.04901523*temp + 10.14333127*humidity;
        heatIndex = heatIndex - .22475541*temp*humidity - .00683783*temp*temp;
        heatIndex = heatIndex - .05481717*humidity*humidity +
        .00122874*temp*temp*humidity;
        heatIndex = heatIndex + .00085282*temp*humidity*humidity -
        .00000199*temp*temp*humidity*humidity;
    } else {
        heatIndex = 0.5 * (temp + 61.0 + ((temp - 68.0)*1.2) + (humidity *
0.094));
    }

    if (humidity < 13 && 80 <= temp <= 112) {
        float adjustment = ((13-humidity)/4) * sqrt((17-abs(temp-95.))/17);
        heatIndex = heatIndex - adjustment;
    }

    return heatIndex;
}

/***** START SET COLOR *****/
void setColor(int inR, int inG, int inB) {
    analogWrite(redPin, inR);
    analogWrite(greenPin, inG);
    analogWrite(bluePin, inB);

    Serial.println("Setting LEDs:");
}

```

```

    Serial.print("r: ");
    Serial.print(inR);
    Serial.print(", g: ");
    Serial.print(inG);
    Serial.print(", b: ");
    Serial.println(inB);
}

/***** START
RECONNECT*****/
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(SENSORNAME, mqtt_user, mqtt_password)) {
            Serial.println("connected");
            client.subscribe(light_set_topic);
            setColor(0, 0, 0);
            sendState();
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

/***** START CHECK SENSOR
*****/
bool checkBoundSensor(float newValue, float prevValue, float maxDiff) {
    return newValue < prevValue - maxDiff || newValue > prevValue + maxDiff;
}

/***** START MAIN
LOOP*****/
void loop() {

    ArduinoOTA.handle();

    if (!client.connected()) {
        // reconnect();
        software_Reset();
    }
    client.loop();

    if (!inFade) {

        float newTempValue = dht.readTemperature(true); //to use celsius remove
the true text inside the parentheses
        float newHumValue = dht.readHumidity();

        //PIR CODE
        pirValue = digitalRead(PIRPIN); //read state of the

```

```

    if (pirValue == LOW && pirStatus != 1) {
        motionStatus = "OFF";
        sendState();
        pirStatus = 1;
    }

    else if (pirValue == HIGH && pirStatus != 2) {
        motionStatus = "ON";
        sendState();
        pirStatus = 2;
    }

    delay(100);

    if (checkBoundSensor(newTempValue, tempValue, diffTEMP)) {
        tempValue = newTempValue;
        sendState();
    }

    if (checkBoundSensor(newHumValue, humValue, diffHUM)) {
        humValue = newHumValue;
        sendState();
    }

    int newLDR = analogRead(LDRPIN);
    newLDR = map(newLDR, 0, 1023, 1023, 0);
    if (checkBoundSensor(newLDR, LDR, diffLDR)) {
        LDR = newLDR;
        sendState();
    }
}

if (flash) {
    if (startFlash) {
        startFlash = false;
        flashStartTime = millis();
    }

    if ((millis() - flashStartTime) <= flashLength) {
        if ((millis() - flashStartTime) % 1000 <= 500) {
            setColor(flashRed, flashGreen, flashBlue);
        }
        else {
            setColor(0, 0, 0);
            // If you'd prefer the flashing to happen "on top of"
            // the current color, uncomment the next line.
            // setColor(realRed, realGreen, realBlue);
        }
    }
    else {
        flash = false;
        setColor(realRed, realGreen, realBlue);
    }
}

if (startFade) {
    // If we don't want to fade, skip it.
    if (transitionTime == 0) {

```

```

        setColor(realRed, realGreen, realBlue);

        redVal = realRed;
        grnVal = realGreen;
        bluVal = realBlue;

        startFade = false;
    }
    else {
        loopCount = 0;
        stepR = calculateStep(redVal, realRed);
        stepG = calculateStep(grnVal, realGreen);
        stepB = calculateStep(bluVal, realBlue);

        inFade = true;
    }
}

if (inFade) {
    startFade = false;
    unsigned long now = millis();
    if (now - lastLoop > transitionTime) {
        if (loopCount <= 1020) {
            lastLoop = now;

            redVal = calculateVal(stepR, redVal, loopCount);
            grnVal = calculateVal(stepG, grnVal, loopCount);
            bluVal = calculateVal(stepB, bluVal, loopCount);

            setColor(redVal, grnVal, bluVal); // Write current values to LED
pins

            Serial.print("Loop count: ");
            Serial.println(loopCount);
            loopCount++;
        }
        else {
            inFade = false;
        }
    }
}
}

/***** START TRANSITION FADER *****/
// From https://www.arduino.cc/en/Tutorial/ColorCrossfader
/* BELOW THIS LINE IS THE MATH -- YOU SHOULDN'T NEED TO CHANGE THIS FOR THE
BASICS
The program works like this:
Imagine a crossfade that moves the red LED from 0-10,
the green from 0-5, and the blue from 10 to 7, in
ten steps.
We'd want to count the 10 steps and increase or
decrease color values in evenly stepped increments.
Imagine a + indicates raising a value by 1, and a -
equals lowering it. Our 10 step fade would look like:
1 2 3 4 5 6 7 8 9 10
R + + + + + + + + + +

```

```

G   +   +   +   +   +
B   -   -   -
The red rises from 0 to 10 in ten steps, the green from
0-5 in 5 steps, and the blue falls from 10 to 7 in three steps.
In the real program, the color percentages are converted to
0-255 values, and there are 1020 steps (255*4).
To figure out how big a step there should be between one up- or
down-tick of one of the LED values, we call calculateStep(),
which calculates the absolute gap between the start and end values,
and then divides that gap by 1020 to determine the size of the step
between adjustments in the value.
*/
int calculateStep(int prevValue, int endValue) {
    int step = endValue - prevValue; // What's the overall gap?
    if (step) {                       // If its non-zero,
        step = 1020 / step;           // divide by 1020
    }

    return step;
}

/* The next function is calculateVal. When the loop value, i,
   reaches the step size appropriate for one of the
   colors, it increases or decreases the value of that color by 1.
   (R, G, and B are each calculated separately.)
*/
int calculateVal(int step, int val, int i) {
    if ((step) && i % step == 0) { // If step is non-zero and its time to
change a value,
        if (step > 0) {           // increment the value if step is
positive...
            val += 1;
        }
        else if (step < 0) {      // ...or decrement it if step is
negative
            val -= 1;
        }
    }
    // Defensive driving: make sure val stays in the range 0-255
    if (val > 255) {
        val = 255;
    }
    else if (val < 0) {
        val = 0;
    }

    return val;
}
/****reset****/
void software_Reset() // Restarts program from beginning but does not reset
the peripherals and registers
{
    Serial.print("resetting");
    ESP.reset();
}

```

6.3 Arduino code rgb controller

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <PubSubClient.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ArduinoJson.h>

/***** WIFI and MQTT INFORMATION (CHANGE THESE FOR YOUR SETUP)
*****/
#define wifi_ssid "aawa" //type your WIFI information inside the quotes
#define wifi_password "waartsalfons36"
#define mqtt_server "192.168.0.50"
#define mqtt_user "lian"
#define mqtt_password "aarts"
#define mqtt_port 1883

/***** MQTT TOPICS (change these topics as you wish)
*****/
#define light_state_topic "bruh/sensornode1"
#define light_set_topic "bruh/sensornode1/set"

const char* on_cmd = "ON";
const char* off_cmd = "OFF";

/***** FOR OTA
*****/
#define SENSORNAME "sensornode1"
#define OTApasword "YouPassword" // change this to whatever password you
want to use when you upload OTA
int OTAport = 8266;

/***** PIN DEFINITIONS
*****/
const int redPin = D1;
const int greenPin = D2;
const int bluePin = D3;
#define PIRPIN D5
#define DHTPIN D7
#define DHTTYPE DHT22
#define LDRPIN A0
```

```

/***** SENSOR DEFINITIONS *****/
/*****/
float ldrValue;
int LDR;
float calcLDR;
float diffLDR = 25;

float diffTEMP = 0.2;
float tempValue;

float diffHUM = 1;
float humValue;

int pirValue;
int pirStatus;
String motionStatus;

char message_buff[100];

int calibrationTime = 0;

const int BUFFER_SIZE = 300;

#define MQTT_MAX_PACKET_SIZE 512

/***** GLOBALS for fade/flash *****/
/*****/
byte red = 255;
byte green = 255;
byte blue = 255;
byte brightness = 255;

byte realRed = 0;
byte realGreen = 0;
byte realBlue = 0;

bool stateOn = false;

bool startFade = false;
unsigned long lastLoop = 0;
int transitionTime = 0;
bool inFade = false;
int loopCount = 0;
int stepR, stepG, stepB;
int redVal, grnVal, bluVal;

bool flash = false;
bool startFlash = false;
int flashLength = 0;
unsigned long flashStartTime = 0;

```

```

byte flashRed = red;
byte flashGreen = green;
byte flashBlue = blue;
byte flashBrightness = brightness;

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

/***** START
SETUP*****/
void setup() {

    Serial.begin(115200);

    pinMode(PIRPIN, INPUT);
    pinMode(DHTPIN, INPUT);
    pinMode(LDRPIN, INPUT);

    Serial.begin(115200);
    delay(10);

    ArduinoOTA.setPort(OTAport);

    ArduinoOTA.setHostname(SENSORNAME);

    ArduinoOTA.setPassword((const char *)OTApasword);

    Serial.print("calibrating sensor ");
    for (int i = 0; i < calibrationTime; i++) {
        Serial.print(".");
        delay(1000);
    }

    Serial.println("Starting Node named " + String(SENSORNAME));

    setup_wifi();

    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);

    ArduinoOTA.onStart([]() {
        Serial.println("Starting");
    });
    ArduinoOTA.onEnd([]() {
        Serial.println("\nEnd");
    });

```

```

    ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
        Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    });
    ArduinoOTA.onError([](ota_error_t error) {
        Serial.printf("Error[%u]: ", error);
        if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
        else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
        else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
        else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
        else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });
    ArduinoOTA.begin();
    Serial.println("Ready");
    Serial.print("IPess: ");
    Serial.println(WiFi.localIP());
    reconnect();
}

```

```

/***** START SETUP
WIFI*****/
void setup_wifi() {

    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);

    WiFi.mode(WIFI_STA);
    WiFi.begin(wifi_ssid, wifi_password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

```

```

/***** START
CALLBACK*****/
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
}

```

```

char message[length + 1];
for (int i = 0; i < length; i++) {
    message[i] = (char)payload[i];
}
message[length] = '\0';
Serial.println(message);

if (!processJson(message)) {
    return;
}

if (stateOn) {
    // Update lights
    realRed = map(red, 0, 255, 0, brightness);
    realGreen = map(green, 0, 255, 0, brightness);
    realBlue = map(blue, 0, 255, 0, brightness);
}
else {
    realRed = 0;
    realGreen = 0;
    realBlue = 0;
}

startFade = true;
inFade = false; // Kill the current fade

sendState();
}

/***** START PROCESS
JSON*****/
bool processJson(char* message) {
    StaticJsonBuffer<BUFFER_SIZE> jsonBuffer;

    JsonObject& root = jsonBuffer.parseObject(message);

    if (!root.success()) {
        Serial.println("parseObject() failed");
        return false;
    }

    if (root.containsKey("state")) {
        if (strcmp(root["state"], on_cmd) == 0) {
            stateOn = true;
        }
        else if (strcmp(root["state"], off_cmd) == 0) {
            stateOn = false;
        }
    }

    // If "flash" is included, treat RGB and brightness differently

```

```

if (root.containsKey("flash")) {
    flashLength = (int)root["flash"] * 1000;

    if (root.containsKey("brightness")) {
        flashBrightness = root["brightness"];
    }
    else {
        flashBrightness = brightness;
    }

    if (root.containsKey("color")) {
        flashRed = root["color"]["r"];
        flashGreen = root["color"]["g"];
        flashBlue = root["color"]["b"];
    }
    else {
        flashRed = red;
        flashGreen = green;
        flashBlue = blue;
    }

    flashRed = map(flashRed, 0, 255, 0, flashBrightness);
    flashGreen = map(flashGreen, 0, 255, 0, flashBrightness);
    flashBlue = map(flashBlue, 0, 255, 0, flashBrightness);

    flash = true;
    startFlash = true;
}
else { // Not flashing
    flash = false;

    if (root.containsKey("color")) {
        red = root["color"]["r"];
        green = root["color"]["g"];
        blue = root["color"]["b"];
    }

    if (root.containsKey("brightness")) {
        brightness = root["brightness"];
    }

    if (root.containsKey("transition")) {
        transitionTime = root["transition"];
    }
    else {
        transitionTime = 0;
    }
}

return true;
}

```

```

/***** START SEND
STATE*****/
void sendState() {
    StaticJsonBuffer<BUFFER_SIZE> jsonBuffer;

    JsonObject& root = jsonBuffer.createObject();

    root["state"] = (stateOn) ? on_cmd : off_cmd;
    JsonObject& color = root.createNestedObject("color");
    color["r"] = red;
    color["g"] = green;
    color["b"] = blue;

    root["brightness"] = brightness;
    root["humidity"] = (String)humValue;
    root["motion"] = (String)motionStatus;
    root["ldr"] = (String)LDR;
    root["temperature"] = (String)tempValue;
    root["heatIndex"] = (String)calculateHeatIndex(humValue, tempValue);

    char buffer[root.measureLength() + 1];
    root.printTo(buffer, sizeof(buffer));

    Serial.println(buffer);
    client.publish(light_state_topic, buffer, true);
}

/*
 * Calculate Heat Index value AKA "Real Feel"
 * NOAA heat index calculations taken from
 * http://www.wpc.ncep.noaa.gov/html/heatindex\_equation.shtml
 */
float calculateHeatIndex(float humidity, float temp) {
    float heatIndex= 0;
    if (temp >= 80) {
        heatIndex = -42.379 + 2.04901523*temp + 10.14333127*humidity;
        heatIndex = heatIndex - .22475541*temp*humidity - .00683783*temp*temp;
        heatIndex = heatIndex - .05481717*humidity*humidity +
        .00122874*temp*temp*humidity;
        heatIndex = heatIndex + .00085282*temp*humidity*humidity -
        .00000199*temp*temp*humidity*humidity;
    } else {
        heatIndex = 0.5 * (temp + 61.0 + ((temp - 68.0)*1.2) + (humidity *
0.094));
    }

    if (humidity < 13 && 80 <= temp <= 112) {
        float adjustment = ((13-humidity)/4) * sqrt((17-abs(temp-95.))/17);
        heatIndex = heatIndex - adjustment;
    }
}

```

```

    }

    return heatIndex;
}

/***** START SET COLOR
*****/
void setColor(int inR, int inG, int inB) {
    analogWrite(redPin, inR);
    analogWrite(greenPin, inG);
    analogWrite(bluePin, inB);

    Serial.println("Setting LEDs:");
    Serial.print("r: ");
    Serial.print(inR);
    Serial.print(", g: ");
    Serial.print(inG);
    Serial.print(", b: ");
    Serial.println(inB);
}

/***** START
RECONNECT*****/
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(SENSORNAME, mqtt_user, mqtt_password)) {
            Serial.println("connected");
            client.subscribe(light_set_topic);
            setColor(0, 0, 0);
            sendState();
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

/***** START CHECK SENSOR
*****/
bool checkBoundSensor(float newValue, float prevValue, float maxDiff) {
    return newValue < prevValue - maxDiff || newValue > prevValue + maxDiff;
}

```

```

/***** START MAIN
LOOP*****/
void loop() {

    ArduinoOTA.handle();

    if (!client.connected()) {
        // reconnect();
        software_Reset();
    }
    client.loop();

    if (!inFade) {

        float newTempValue = dht.readTemperature(true); //to use celsius remove
the true text inside the parentheses
        float newHumValue = dht.readHumidity();

        //PIR CODE
        pirValue = digitalRead(PIRPIN); //read state of the

        if (pirValue == LOW && pirStatus != 1) {
            motionStatus = "OFF";
            sendState();
            pirStatus = 1;
        }

        else if (pirValue == HIGH && pirStatus != 2) {
            motionStatus = "ON";
            sendState();
            pirStatus = 2;
        }

        delay(100);

        if (checkBoundSensor(newTempValue, tempValue, diffTEMP)) {
            tempValue = newTempValue;
            sendState();
        }

        if (checkBoundSensor(newHumValue, humValue, diffHUM)) {
            humValue = newHumValue;
            sendState();
        }

        int newLDR = analogRead(LDRPIN);
        newLDR = map(newLDR, 0, 1023, 1023, 0);
        if (checkBoundSensor(newLDR, LDR, diffLDR)) {
            LDR = newLDR;
            sendState();
        }
    }
}

```

```

    }

}

if (flash) {
    if (startFlash) {
        startFlash = false;
        flashStartTime = millis();
    }

    if ((millis() - flashStartTime) <= flashLength) {
        if ((millis() - flashStartTime) % 1000 <= 500) {
            setColor(flashRed, flashGreen, flashBlue);
        }
        else {
            setColor(0, 0, 0);
            // If you'd prefer the flashing to happen "on top of"
            // the current color, uncomment the next line.
            // setColor(realRed, realGreen, realBlue);
        }
    }
    else {
        flash = false;
        setColor(realRed, realGreen, realBlue);
    }
}

if (startFade) {
    // If we don't want to fade, skip it.
    if (transitionTime == 0) {
        setColor(realRed, realGreen, realBlue);

        redVal = realRed;
        grnVal = realGreen;
        bluVal = realBlue;

        startFade = false;
    }
    else {
        loopCount = 0;
        stepR = calculateStep(redVal, realRed);
        stepG = calculateStep(grnVal, realGreen);
        stepB = calculateStep(bluVal, realBlue);

        inFade = true;
    }
}

if (inFade) {
    startFade = false;
    unsigned long now = millis();
    if (now - lastLoop > transitionTime) {
        if (loopCount <= 1020) {

```

```

        lastLoop = now;

        redVal = calculateVal(stepR, redVal, loopCount);
        grnVal = calculateVal(stepG, grnVal, loopCount);
        bluVal = calculateVal(stepB, bluVal, loopCount);

        setColor(redVal, grnVal, bluVal); // Write current values to LED
pins

        Serial.print("Loop count: ");
        Serial.println(loopCount);
        loopCount++;
    }
    else {
        inFade = false;
    }
}
}
}

/***** START TRANSITION FADER
*****/
// From https://www.arduino.cc/en/Tutorial/ColorCrossfader
/* BELOW THIS LINE IS THE MATH -- YOU SHOULDN'T NEED TO CHANGE THIS FOR THE
BASICS

The program works like this:
Imagine a crossfade that moves the red LED from 0-10,
the green from 0-5, and the blue from 10 to 7, in
ten steps.

We'd want to count the 10 steps and increase or
decrease color values in evenly stepped increments.
Imagine a + indicates raising a value by 1, and a -
equals lowering it. Our 10 step fade would look like:
1 2 3 4 5 6 7 8 9 10
R + + + + + + + + + +
G   +   +   +   +   +
B     -     -     -

The red rises from 0 to 10 in ten steps, the green from
0-5 in 5 steps, and the blue falls from 10 to 7 in three steps.
In the real program, the color percentages are converted to
0-255 values, and there are 1020 steps (255*4).
To figure out how big a step there should be between one up- or
down-tick of one of the LED values, we call calculateStep(),
which calculates the absolute gap between the start and end values,
and then divides that gap by 1020 to determine the size of the step
between adjustments in the value.
*/
int calculateStep(int prevValue, int endValue) {
    int step = endValue - prevValue; // What's the overall gap?
    if (step) {                       // If its non-zero,

```

```

    step = 1020 / step;          // divide by 1020
}

return step;
}

/* The next function is calculateVal. When the loop value, i,
   reaches the step size appropriate for one of the
   colors, it increases or decreases the value of that color by 1.
   (R, G, and B are each calculated separately.)
*/
int calculateVal(int step, int val, int i) {
    if ((step) && i % step == 0) { // If step is non-zero and its time to
change a value,
        if (step > 0) {           // increment the value if step is
positive...
            val += 1;
        }
        else if (step < 0) {      // ...or decrement it if step is
negative
            val -= 1;
        }
    }

    // Defensive driving: make sure val stays in the range 0-255
    if (val > 255) {
        val = 255;
    }
    else if (val < 0) {
        val = 0;
    }

    return val;
}

/****reset****/
void software_Reset() // Restarts program from beginning but does not reset
the peripherals and registers
{
    Serial.print("resetting");
    ESP.reset();
}

```