**Hoe krijg je mensen meer in beweging met behulp van een embedded systeem?**

Door Ruben Middelman



|  |  |
| --- | --- |
| Student | Ruben Midelman |
| Student nr. | 1740398 |
| Klas | EV1A0.4 |
| Corrector |  |

Het bestuur van de Stichting Hogeschool Utrecht te Utrecht aanvaardt geen enkele aansprakelijkheid voor schade voortvloeiende uit het gebruik van enig gegeven, hulpmiddel, werkwijze of procedure in dit verslag beschreven. Vermenigvuldiging zonder toestemming van de auteur(s), de school of het bedrijf (indien van toepassing) is niet toegestaan.

**Inleiding [223 woorden]**

In deze coronatijd is er erg te merken dat mensen een stuk minder bewegen. Mensen hoeven alleen nog maar achter een stoel te zitten en online in te loggen om op een andere locatie te zijn okal is deze locatie echter niet echt een andere plek maar meer een nieuwe omgeving met andere mensen. Dit zorgt er voor dat veel mensen veel minder bewegen. Volgens het CBS fietst de gemiddelde Nederlander 2,9 Km per dag. Nu kunnen wij met behoorlijk wat zekerheid zeggen dat dit door de corona crisis heel erg naar beneneden is gegaan doordat mensen echter niet meer naar buiten hoeven of zelfs mogen. Dit betekend dat voor de meeste Nederlanders hun dagelijkse beweging enorm gedaald is. Ook is er met onderzoek door het IPOS gezien dat een kwart van de Nederlanders erg is aangekomen sinds de corona periode. Het is dus echt de hoogste tijd dat er meer bewogen wordt in deze tijden vandaar dit onderzoek met als hoofdvraag: Hoe krijg je mensen meer in beweging krijgen met behulp van een embedded systeem?

Voor dit onderzoek wordt dus eerst gekeken naar waarom mensen een stuk minder bewegen in de corona crisis. Daarna wordt er gekeken naar een oplossing op dit probleem met behulp van een embedded systeem. Dan zullen wij als laatste kijken naar hoe het technische gedeelte echt werkt.

De pagina indeling zal er zo uitzien:

1. **De Doelgroep [243 woorden]**

Aan het begin van een onderzoek is het erg belangrijk om te kijken naar wat voorn mensen je probeert te helpen met dit onderzoek. Hiervoor zijn een paar persona’s gemaakt zodat er een meer grafisch beeld is van de doelgroep die geholpen moet worden met behulp van een embedded systeem. Een paar vragen voor de persona’s waren:

* Wat doet u op een dagelijkse basis?
* Wat doet u aan beweging?
* Heeft corona uw beweging gedrag beïnvloed? Zo ja, hoe?
* Wat doet u om te relaxen?
* Viel sporten eerst onder relaxen maar is door de coronaperiode dat nu veranderd?
* Wat voorn electronische apparaten gebruikt u op een dagelijkse -basis?

Uit deze persona’s (die onderaan dit verslag zijn toegevoegd) kwamen een paar mooie verschijnselen:

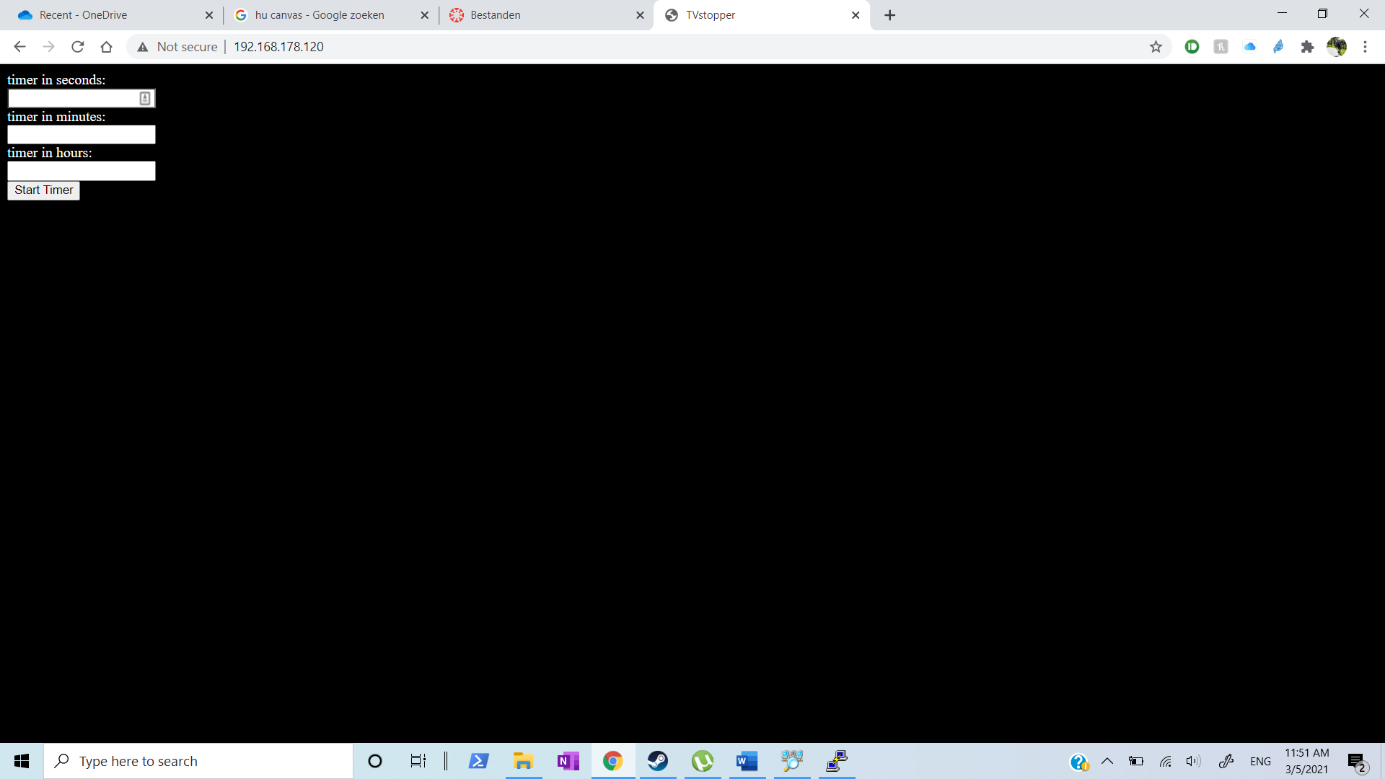
* Mijn doelgroep is jong volwassenen van de leeftijd tussen de 18 en 22.
* Deze jong volwassenen studeren nog op de middelbare school of doen aan hoger onderwijs.
* Deze jong volwassenen fietste eerst altijd naar hun school maar door school sluiting fietsen zij een stuk minder.
* De jongeren houden er van om te relaxen met TV of video games.
* Ook willen deze jongeren meer bewegen maar zijn ze meestal te vermoeid na school om echt te bewegen dus gaan ze sneller achter een TV zitten.
* De doelgroep is niet zeer technisch maar kan met basis electronica omgaan. Als iets wordt voorgedaan kunnen ze het nadoen.

Dit schets een behoorlijk goed beeld wat nodig is om een goed product voor de doelgroep te maken.

1. **Aanpak van het onderzoek [307 woorden]**

Voor dit onderzoek is gekeken naar hoe mensen meer in beweging kunnen komen. Uit het onderzoek bleek dat mensen na een hele dag achter een computer zitten te moe zijn om nog echt beweging te doen. Mensen gaan liever ontspannen achter een tv zitten wat eigenlijk het tegenovergestelde is van bewegen. Mensen willen wel sporten maar door het constant moe zijn van de online lessen is er niet echt genoeg motivatie. Hier zit dus het probleem wat wij willen oplossen met dit project. Hoe kunnen we mensen weer gemotiveerd krijgen om te bewegen. Als we dan ook kijken naar wat er uit de persona’s bleek, dan zien we dat mensen wel willen bewegen maar ook willen ze relaxen. Dit betekend dus dat mensen wel willen sporten als er ook tijd is om te relaxen daarna. Toen werd er naar gekeken wat de beste manier is om mensen te motiveren. Dit kunnen erg veel manieren zijn. Als je naar verschillende doelgroepen kijkt zie je dat mensen beide op positiefe en negatiefe manier gemotiveerd kunnen worden. Een voorbeeld voor positiefe motivatie is als je een goede situatie hebt en door de motivatie wordt deze situatie nog beter. Voor negatiefe motivatie moet je denken dat je een probleem hebt dit probleem moet worden opgelost wat je motiveert om het optellossen. Hierdoor kwam het idee om een machine te maken die je vraagt om een timer in te stellen zodat je na een bepaalde tijd wel gedwongen bent om te bewegen. Dit geeft je motivatie om te sporten zodat je daarna weer kan relaxen. Dit zorgt er voor dat er beide positiefe en negatieve motivatie wordt meegenomen. Het positieve gedeelte is de gedachten dat je straks klaar bent en dan kan relaxen. Het negatieve gedeelte is dat je gedwongen wordt om te sporten wat niet een heel erg fijn gevoel geeft.

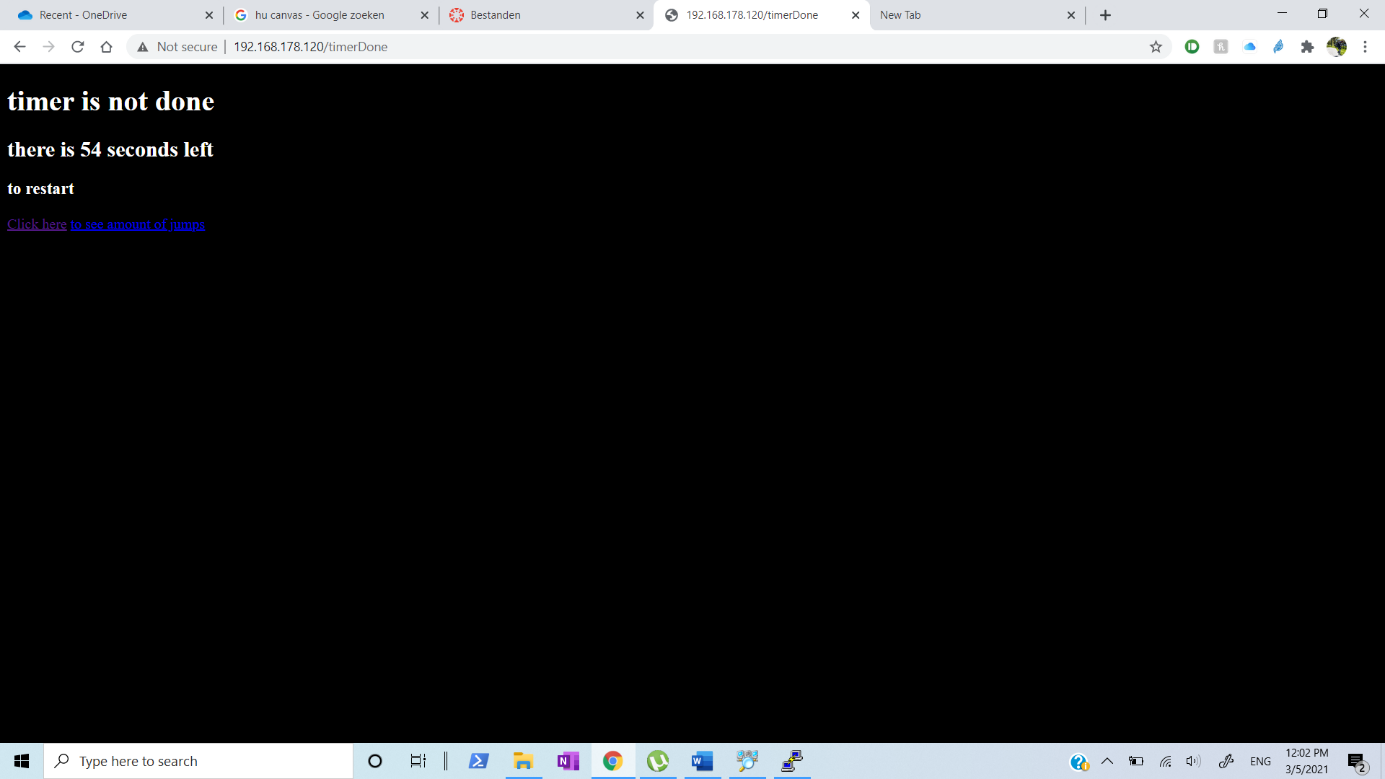
1. **Toegepaste principes**
2. **Accecibility [185 woorden]**

Volgens het principe accecibility is in een design het gebruik het belangrijkst. Daarom moet het gebruik zo makkelijk mogelijk gaan. Hierdoor kwam mijn idee om een webapp te maken. Dit is in mijn ogen een van de makkelijkste manieren om een goede interface te maken waar iedereen makkelijk mee kan werken. Echter is het wel lastig om een goede webapp te maken. 

Hier is een foto van het eerste scherm wat je ziet als je de webapp opent. Zoals je ziet is het erg makkelijk te besturen want het enige wat wordt gevraagd is om de tijd in te stellen. Er hoeft zelfs niet nagedacht worden over hoe je de tijd opschrijft. Als er maar een schrijf veld zou zijn zou de gebruiker moeten twijfelen over hoe hij/zij de tijd zou moeten opschrijven wat een stuk meer errors zou geven. Nu is er alleen maar de mogelijkheid om 3 getallen in te vullen wat er voor zorgt dat de errors minimaal zijn (er is nog wel de kans dat een gebruiker tekst invult wat er voor zorgt dat de server een error geeft.)

1. **Occoms razor [200 woorden]**

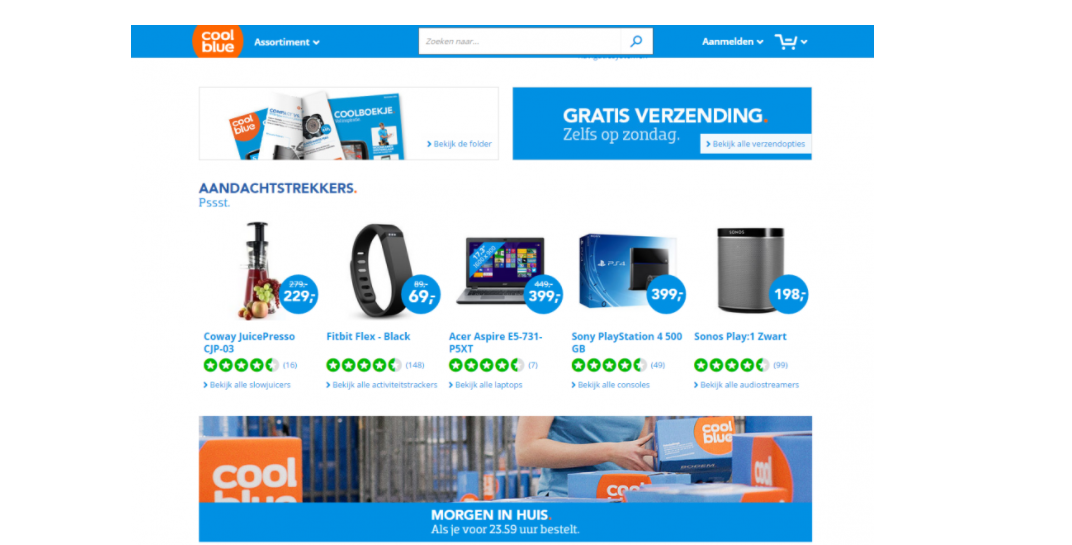
Occoms razor zegt hou het zo makkelijk mogelijk. Dit is op meerdere manieren toegepast. Als je naar de webapp kijkt zie je dat er niet veel mogelijkheid is om dingen aan te passen. Dit is zodat er geen over gecompliceerde dingen mogelijk zijn. Ook staat er zo min mogelijk tekst op het scherm zodat de

gebruiker niet te veel info in een keer krijgt

Als je bijvoorbeeld kijkt naar deze pagina zie je dat er alleen maar opstaat of de timer al klaar is en er staat nog op hoeveel seconde er over zijn. Dit zorgt er voor dat er geen overvloed van info op een pagina staat.

Ook heb is occoms razor toegepast op basis van programmeren. Mijn eerste idee was om te meten hoe vaak er wordt gesprongen met een een *teensy* die met bluetooth gekoppeld is met de *Raspberry pi* die de webapp host. Toen werd er ontdekt dat de connectie tussen *raspberry pi* en een bluetooth module best wel lastig is een niet ging lukken in de gegeven tijd. Toen is er gedacht aan occoms razor, dat zorgde er voor dat er gedacht werd aan de makkelijkste optie. Dat was natuurlijk gewoon de *teensy* rechtstreeks in de *raspberry pi* te pluggen en de data te versturen via Seriële communicatie.

1. **Color [260 woorden]**

Volgens het boek is kleur in interface erg belangrijk. Ook moest (volgens occoms razor) de webapp simpel en makkelijk houden. Volgens color is het dan belangrijk om niet te aandacht trekkende kleuren te gebruiken. Denk bij dit soort kleuren aan neon kleuren als neon geel of groen. Ook is er online wat onderzoek gedaan. Daar werden dingen over kleuren gezegd die heel erg interessant zijn. Kijk bijvoorbeeld naar deze pagina van coolblue. Hier is alles blauw. De blauwe kleur geeft een relaxend gevoel wat heel erg fijn is tijdens het winkelen

Ook stond er online dat zwart een fijne kleur is als je probeert te lezen en als je je probeert te concentreren. Daarom is er voor gekozen om mijn webapp de kleur wit op zwart te geven. Dit zorgt er voor dat de pagina niet leeg voelt terwijl hij wel leeg is. Ook zorgt het er voor dat de tekst erg fijn te lezen is. Echter is er een ding wat niet is kunnen aanpassen. De kleur die ene link heeft is in de basis taal die altijd gebruikt wordt blauw. Dit zorgt ervoor dat de tekst voor de linken blauw op zwart is. Dit is iets wat in later projecten meer uitgezocht moet worden. Ook wordt er met color gekeken naar de ledstrip die onder de TV zit. Die veranderd van groen naar rood. Dit zijn basis kleuren die wij als mensen heel erg herkenen (denk aan stoplichten). Daarom is er ook voor deze kleuren gekozen. Wij als mensen associëren groen met doorgaan en rood met stoppen.

1. **Techniek [1019 woorden]**

Laten we een klein kijkje nemen in de code. Er is groot deel om over te gaan. Laten we beginnen met het makkelijkste. De Spring counter.

Dit is het enige stuk code wat in C is geschreven dit wordt namelijk gedraaid op een *teensy 4.0.*

**void** **loop**() {

// makes a state change finder so that i counts how mant times a light disapears from the LDR so that ou kan count the jumping jacks

sensorVal = analogRead(sensorPin);

digitalWrite(ledPin, LOW);

startSensorVal = sensorVal;

delay(**50**);

sensorVal = analogRead(sensorPin);

secondSensorVal = sensorVal;

checkerVal = startSensorVal - secondSensorVal;

**if**(checkerVal > **100**){

counter++;

Serial.println(counter);//data that is being Sent

}

Dit is het enige stukje code wat echt wat belangrijks doet in de *Teensy.* Er staat namelijk een laser op een LDR gericht. De LDR wordt uitgelezen en dan wordt het 50 milliseconden later weer uitgelezen zodat er een verschil gemeten kan worden. Deze waarde worden dan tegen elkaar gezet zodat er gezien kan worden of er een verschil is. Als het verschil groter is dan 100 dan wordt er wordt er 1 toegevoegd aan de counter. Dit betekend dat er een keer gesprongen is. Dit wordt vervolgens doorgestuurd via seriële communicatie naar de *raspberry pi.*

Laten we nu kijken naar wat er echt gebeurt in de *raspberry pi.*

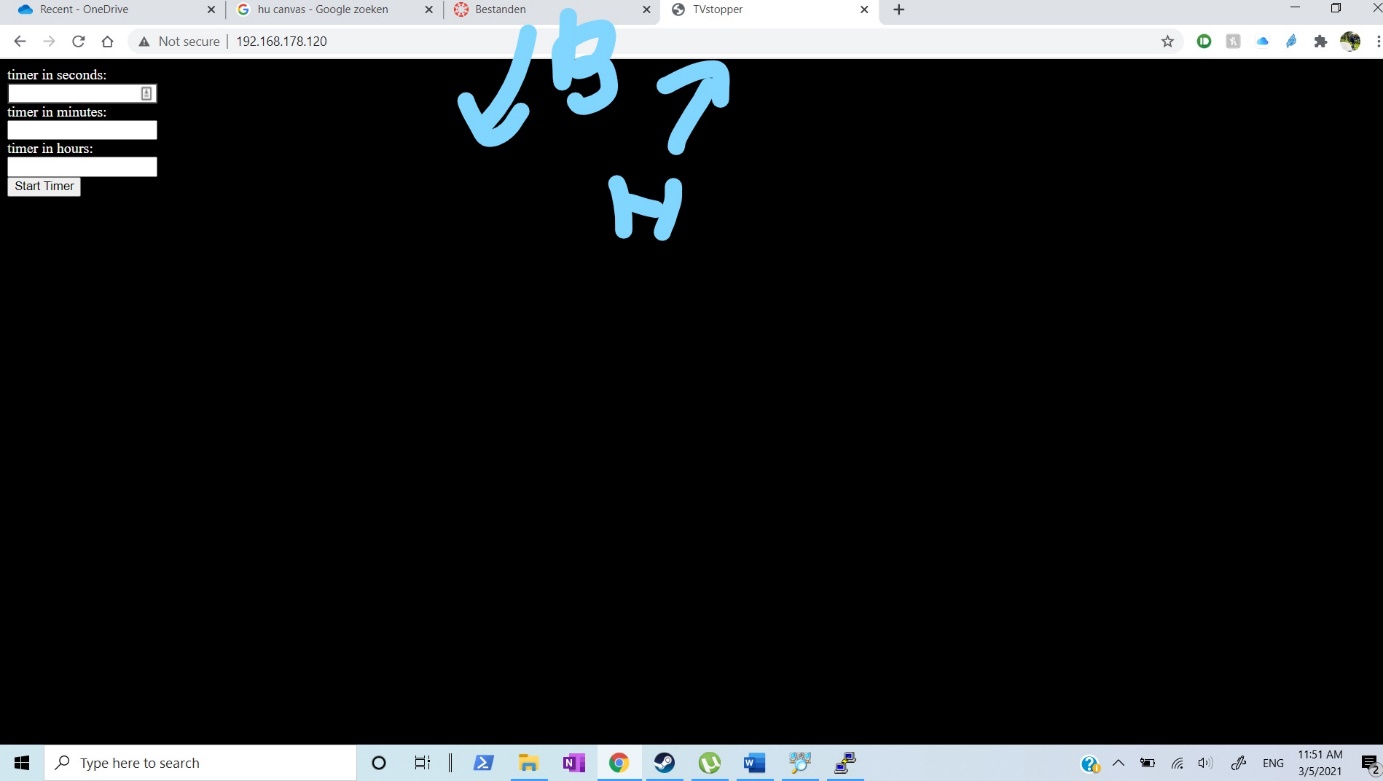
Dit is het eerste stukje basiscode van de *raspberry pi* die een webserver lanceert.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | app = Flask(\_\_name\_\_)  **@app**.route("/")  **def** **hello**():  templateData = {  'title' : 'TVstopper',  }  **return** render\_template('index.html', \*\*templateData) |

Dit stuk python is deel van een groot python bestand wat een webserver op het ip van de raspberry pi maakt. Om precies te zijn start het een webserver op het adres van de raspberry pi met een / er achter. De server zelf wordt echter later in de code pas opgestart maar daar later meer over. Als de server dan een get request krijgt (een get request is een vraag naar de web pagina) dan laad hij deze functie. Dit is maar een beetje een voorbeeld functie want het geeft alleen de titel van de webpagina mee. Daarna staat er als het ip van de raspberry pi met een / wordt gevraagd laad dan de index.html file en stuur de data mee.

Dan nemen we een kijkje in de HTML file om te zien wat daar gebeurt.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <!DOCTYPE html>  <head>  <title>{{ title }}</title>  <link rel="stylesheet" href="../static/style.css/">  </head>  <body>  <form action="{{handle\_data}}" method="post">  <label for="tSec">timer in seconds:</label><br>  <input type="text" id="tSec" name="tSec" value=""><br>  <label for="tMin">timer in minutes:</label><br>  <input type="text" id="tMin" name="tMin" value=""><br>  <label for="tHour">timer in hours:</label><br>  <input type="text" id="tHour" name="tHour" value=""><br>  <button type="submit">Start Timer</button>  </form>  </body>  </html> |

Dit is de basis HTML code voor de home pagina. Bovenin start de pagina op daar wordt over de head en de body gepraat. 

Als er wordt gepraat over een head en een body dan hebben we het over de gedeeltes waar een H en een B bij staan. De Header is dus het ding wat boven de pagina staat en de Body is eigenlijk de rest van de pagina. In het head gedeelte staat de naam van de pagina en welk CSS bestand er gebruikt moest worden. Hier wordt later op terug gekomen. Onder in het body gedeelte staat eigenlijk alleen maar: maak 3 tekst input velden aan en maak een knop aan zodat alles gesubmit kan worden. De webpagina is erg beïnvloed door occoms razor. Het moest zo simpel mogelijk. Ook vallen de kleuren een beetje op. Dit is gebasseerd op het principe color. Als je nu namelijk naar de webpagina kijkt is het leeg en niet druk maar het voelt niet leeg. Als je kijkt naar hoe de kleuren gedaan zijn zullen we erg snel CSS er bij halen. CSS wordt gebruikt om webpagina’s mooier te maken. Voor dit project is CSS gebruikt om de pagina zwart te maken en de woorden wit te maken. Dit wordt zo gedaan

|  |  |
| --- | --- |
| 1  2  3  4 | body {  **background**: black;  **color**: white;  } |

Dit is dus een klein stukje code maar door de kleuren verandert het hele project al.

Echter beginnen de lastige dingen weer in python pas. Kijk bijvoorbeeld naar de code die de LEDStrip timer laat branden.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | **def** **Timer**():  **global** timerState  **global** timeLeft  starttime =time.time()  currenttime= time.time()  x = **0**  timerState = **1**  pixelChangePS = timerTotal / num\_pixels  print(pixelChangePS)  pixelChange = pixelChangePS  **while** timerState == **True**:  **if** timerTotal > (currenttime - starttime):  **if**(starttime + pixelChange) < currenttime:  timeCMS = currenttime - starttime  timeLeft = timerTotal - timeCMS  pixels[x] = (**255**, **0**, **0**)  pixels.show()  x = x+**1**  pixelChange = pixelChange + pixelChangePS  print(timeLeft)  currenttime = time.time()  **else**:  timeLeft = **0**  TurnOffTV()  print("timer is done")  pixels.fill((**255**,**0**,**0**))  pixels.show()  time.sleep(**0.25**)  pixels.fill((**0**,**0**,**0**))  pixels.show()  time.sleep(**0.25**)  pixels.fill((**255**,**0**,**0**))  pixels.show()  timerState = **0** |

Deze functie checkt de hele tijd of de tijd die verstreken is gelijk is aan de tijd die gegeven is (later wordt er besproken hoe deze tijd wordt binnengehaald van de webserver). Deze functie checkt dus constant of de verstreken tijd al groter is dan de tijd die ingepland stond als dat niet zo is dan gaat de loop gewoon door. Als dat wel zo is dan stop de timer en zijn alle ledjes in de LEDstrip groen. Als de verstreken tijd nog niet groter is dan de geplande timer tijd dan wordt er gecheckt of er al ledjes van groen naar rood moeten veranderen. Dit zorgt er voor dat de timer geleidelijk naar beneden gaat in plaats van dat de timer gelijk van groen naar rood wat best wel een belangrijke feature. Als dan de timer af is word een nieuwe functie ingezet die de TV uit doet. Dit stuurt gewoon een bericht naar de arduino dat de hij een IR bericht moet uitsturen.

Eigenlijk is hier alles heel erg logisch en simpel geprogrammeerd. Er was echter een probleem. Veel van deze code moet simultaan draaien. Anders moet de webapp stopen met draaien totdat de timer functie klaar is. Dit is echter niet fijn gezien dat de doelgroep te snel zal wegdrukken in plaats van op de code te wachten. Daar is dit idee dus op gekomen.

|  |  |
| --- | --- |
| 1  2  3  4 | timerThread = threading.Thread(target = Timer)  webThread = threading.Thread(target = app.run(host='0.0.0.0', port=**80**, debug=**True**))  webThread.start() |

Er wordt multithreading gebruikt zodat de code simultaan kan runnen. Dit lost dus het probleem op van het wachten en zorgt er voor dat de doelgroep de code goed kan gebruiken.

1. **Bronnen**

*Arduino IR LED*. (z.d.). www.circuitbasics.com. Geraadpleegd op 1 maart 2021, van https://elektronicavoorjou.nl/arduino-project-basis-ir-remote-control/

*Code Naar Word*. (z.d.). http://hilite.me/. Geraadpleegd op 5 maart 2021, van http://hilite.me/

*Intro to flask*. (z.d.). https://www.fullstackpython.com/. <https://www.fullstackpython.com/flask.html>

*Linkdin learning HTML course*. (z.d.). https://www.linkedin.com/learning/html-essential-training-4. Geraadpleegd op 20 februari 2021, van https://www.linkedin.com/learning/html-essential-training-4

Code en persona’s te vinden op https://github.com/rubenmiddelman/Systeem3C.git