Projectdossier Embedded Linux

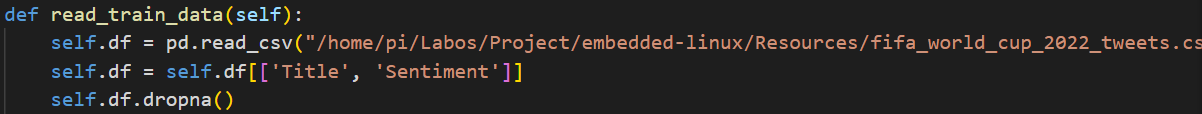
Sentiment analyse op real life nieuwstitels

# Abstract

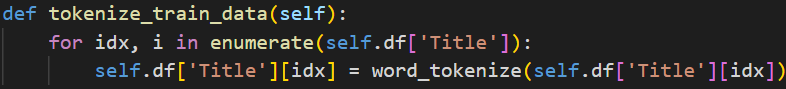
De Raspberry Pi draait via Apache een webserver, die enkel https toelaat en beveiligt is met een gebruikersnaam en wachtwoord. Op deze html pagina kan gedrukt worden op een knop waarna er via de News API nieuwsartikelen omtrent Bitcoin worden ingelezen. Er wordt sentiment analyse toegepast op deze titels via NLP (Natural Language Processing). Zo wordt bepaald of ze positief, neutraal of negatief zijn. Verder worden de ingelezen titels per sentiment gesorteerd en uitgeprint op de HTML-pagina. Ook worden de titels per sentiment weggeschreven naar de cloud. Verder zal de Raspberry Pi communiceren via BLE (Bluetooth Low Energy) met een Arduino Nano 33 BLE. Deze arduino zal op zijn beurt via I2C een OLED-display aansturen. Op deze display wordt het aantal titels per sentiment afgebeeld. Tenslotte wordt er nog via MQTT een melding gestuurd naar een ESP32 die een ledje laat branden indien er nieuwe titels zijn ingelezen.

# NLP-model

Om de sentiment van de titels te bepalen wordt gebruik gemaakt van een klassiek lineair classificatie model. Dit model moet eerst getraind worden met een dataset van voorbeelden. Een csv-bestand genaamd “fifa\_world\_cup\_2022\_titels.csv” werd van het internet gedownload. Dit bestand bevat real life titels over het recente WK voetbal. Iedere titel heeft een correcte sentiment toegekregen. Onderstaande code toont aan dat deze file wordt ingelezen en omgezet naar een DataFrame object van de pandas package. Ook worden de lege rijen al verwijdert.



Om het leerproces te vereenvoudigen worden de titels wat opgekuist. Om dit te kunnen doen moeten we van iedere titel (rij in dataframe) een array van woorden maken i.p.v. één lange string. Dit gebeurt via de kant-en-klare “word\_tokenize” van de nltk package. Deze wordt toegepast op iedere rij of dus titel van het dataframe.



Zoals reeds vermeld kan het leerproces sneller gebeuren door eerst de titels wat te wijzigen. Zo worden speciale karakters en Engelse stopwoordjes via “stopwords.words(“english”)” verwijdert. Door te itereren over alle woorden uit alle titels kan gekeken worden of ze voorkomen in de set van stopwoordjes, indien dat zo is wordt dat woord verwijdert uit de array van woorden. Om de speciale karakters eruit te halen, worden ook nog eens alle letters van alle woorden doorlopen. Via de “translate” methode van een string object kunnen de speciale karakters vervangen worden oor een lege char. Ook worden alle woorden naar lower case gezet.

Afbeelding met tekst, schermopname

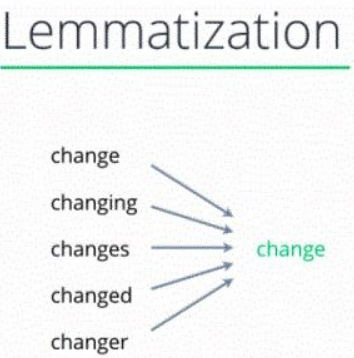
Automatisch gegenereerde beschrijving

Nog een manier om het leerproces te vereenvoudigen en te verbeteren is door de woorden te “lemmatizen”. Deze zal de stam nemen van werkwoorden aangezien vervoegde werkwoorden toch dezelfde achterliggende betekenis hebben. Ook het lemmatizen gebeurt via de kant-en-klare “WordNetLammatizer” van nltk. De “lemmatize” methode wordt namelijk toegepast op ieder woord.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Onderstaand voorbeeld toont de werking van “lemmatizen”.



Nog een laatste stap voor het trainen, is het weer aan elkaar zetten van de titels. Via de “join” methode die toegepast wordt op een lege string kan via de “map” methode weer alle woorden samengevoegd worden. Zo is er weer één lange string voor iedere titel.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Om het model te trainen en te evalueren wordt de grote dataframe met alle titels opgesplitst. De “train\_test\_split” methode van de sklearn package zal at random 30% van de titels apart nemen om het model achteraf te kunnen evalueren. De overige 70% dient voor het trainen. Vooraleer een lineair classificatie model de sentiment van titels kan voorspellen, moeten deze titels gevectoriseerd worden. Dit gebeurt via een “TfidfVectorizer” van de sklearn package. Deze zal maximum 1500 woorden gaan bijhouden. Vervolgens wordt de “vectorizer” getraind. Via een Pipeline object kan dan een lineair classificatie model gemaakt worden. De “vectorizer” wordt gelinkt aan een “LogisticRegression” object van de sklearn package. Nu is het NLP-model compleet en kan dit op zijn beurt gefit of dus getraind worden.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Het “model\_tfidf” object kan voorspellingen maken via de “predict” methode. Om het model objectief te kunnen evalueren wordt de achtergelaten 30% van de titels gebruikt aangezien het lineair classificatie model deze titels nog nooit heeft gezien. De accuracy wordt bepaald via de ingebouwde “accuracy\_score” functie van sklearn.

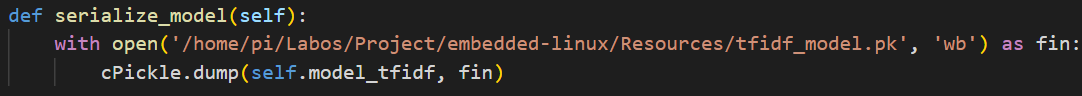
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Het model slaagt erin om op de testdata (de achtergelaten titels) 69,1% accuracy te halen. Hier zou nog plaats zijn voor verbetering, maar dit is een mooi begin. Op de trainingsdata scoort het model typisch beter en dit omdat het model deze data al talloze keren heeft gezien tijdens het trainen.



Aangezien alle bovenstaande code veel rekenwerk vraagt en dus tijd nodig heeft, kan dit script niet elke keer uitgevoerd worden wanneer op de knop gedrukt wordt bij de HTML-pagina. Om dit probleem op te lossen wordt het uiteindelijk “model\_tfidf” object via binaire serialisatie weggeschreven naar een “tfidf\_model.pk” bestand die lokaal op de Pi staat. Dit gebeurt via de “dump" methode van de Pickle package. Het bestand kan dan elke keer bij het drukken van de knop gedeserialiseerd worden.



# Webserver

Om een webserver lokaal te hosten op de Raspberry Pi wordt gebruik gemaakt van Apache. Ook werd PHP reeds geïnstalleerd in labo 2. De HTML, css en php pagina’s worden opgeslagen in de /var/www/html map. Door te browsen naar [https://framboos17.local/titels\_home.html](https://framboos17.local/tweets_home.html) kan deze webpagina bekeken worden.

Via onderstaande code wordt het “update.py” bestand uitgevoerd indien er op de webpagina op de knop werd gedrukt. Dit gebeurt via de “shell\_exec” methode van PHP. Ook worden de eventuele foutmeldingen via “2>&1” op het scherm getoond. De “header” methode zal er voor zorgen dat op het einde van de PHP code de vorige HTLM pagina automatisch opnieuw wordt geladen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Onderstaande code zorgt ervoor dat een form wordt verstuurd indien op de knop wordt gedrukt. Indien de PHP-code een “on” krijgt, wordt het “update.py” bestand uitgevoerd.

Afbeelding met tekst, schermopname, Lettertype, lijn

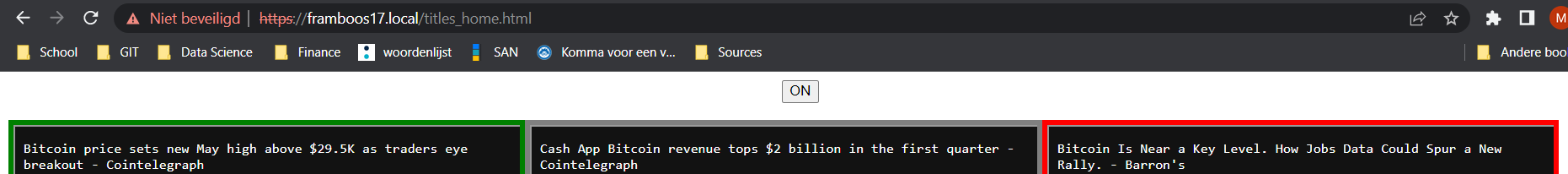
Automatisch gegenereerde beschrijvingtw

Via de “iframe” attributen kan de inhoud van een lokaal bestand worden afgeprint op de HTML-pagina. Dit wordt eenmaal gedaan voor de lokaal opgeslagen titels alsook voor de titels opgeslagen in de cloud. Het lokale “positive\_titels\_cloud.txt”, te zien op onderstaande afbeelding, wordt telkens overschreven met alle titels van de cloud. Dit lokale bestand is slechts een tussenstap om het te kunnen tonen op de webpagina.

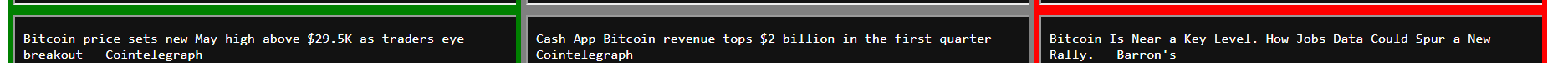
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Dit is een uiteindelijke resultaat van de lokale webpagina. Er is een positieve titel verkeerd gemarkeerd. Deze zou dus eigenlijk in de linker kolom moeten staan. Het model maakt dus zeker fouten bij het voorspellen en dit omdat het een zeer beperkt model is.

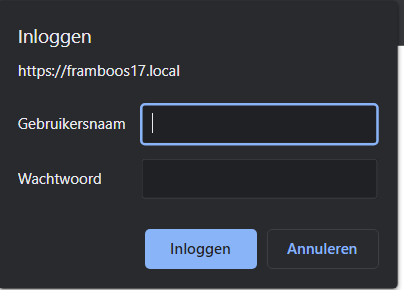


Dit zijn de onderste iframes die de titels lezen vanuit de cloud. Deze komen overeen met de lokaal opgeslagen bestanden.



Via de technieken uit labo 2 is het niet mogelijk om via HTTP te surfen naar de webserver. Afbeelding met tekst

Automatisch gegenereerde beschrijving

Ook zal er gevraagd worden om in te loggen indien de webserver via HTTPS wordt bereikt.

# Update

Zoals reeds vermeld is er één “update.py” bestand die elke keer uitgevoerd wordt bij het drukken van de HTML-knop. Dit bestand stuurt alles aan. Als eerst worden nieuwe titels via de NewsAPI ingelezen. Vervolgens wordt sentiment analyse uitgevoerd op deze titels. Daarna worden de sentiments weggeschreven. Ook wordt er via BLE gecommuniceerd naar de Arduino om het aantal titels per sentiment te updaten. Tenslotte zal de Raspberry Pi via MQTT de ESP32 inlichten.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

## NewsAPI

Via de ‘NewsApiClient’-package, die via NewsAPI beschikbaar werd gezet, kunnen er nieuwsartikelen van veel verschillende landen, bronnen en onderwerpen ingelezen worden. In de constructor werd reeds een client-object aangemaakt met een persoonlijke API-key. Onderstaande methode zal artikelen opvragen omtrent het onderwerp ‘Bitcoin’ die in het Engels zijn geschreven. Vervolgens wordt de titel van elk artikel in een lijst opgeslagen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

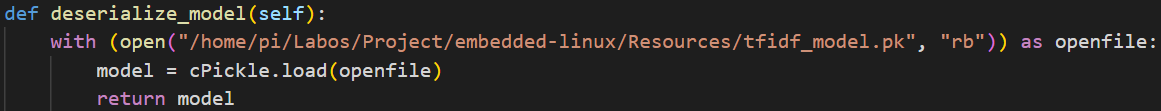
## Sentiment predictions

Omdat het NLP-model, die de predicties maakt, niet elke keer opnieuw opgebouwd kan worden in een korte tijd, wordt deze hergebruikt door te deserialiseren. Onderstaande afbeelding toont aan dat het model elke keer wordt opgevraagd en gebruikt om voorspellingen te maken.

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Via de ingebouwde “open” methode kan het lokaal opgeslagen “tfidf\_model.pk” bestand bekeken worden. Ook wordt opnieuw de Pickle package gebruikt om het te deserialiseren.



Tenslotte wordt opnieuw via de “predict” methode van het NLP object voorspellingen gemaakt. Dit resulteert in een array van strings bv.: [“positive”, “positive”, “negative”…]

## Wegschrijven predictions

### Naar cloud

Als eerst wordt een “cloud.py” object aangemaakt. Deze bevat de nodige info om contact op te zetten met de Influxdb database, alsook een “write” en “read” methode. Een Influxdb database is een time series database, wat eigenlijk niet zo logisch is voor enkel titels met hun sentiment op te slaan. Echter is voor Influxdb gekozen omdat deze gemakkelijk en gratis te gebruiken is.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een Influxdb organisatie (hier genaamd “Embedded Linux”) bestaat uit buckets (= databases) met daarin verschillende measurements (= tabellen). Elke tabel of measurement kan dan verschillende fiels (= kolommen) hebben. De “titels” tabel heeft zoals onderstaande afbeelding aantoont drie kolommen voor elke sentiment.

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Multimediasoftware, Lettertype

Automatisch gegenereerde beschrijving

Ook is er een acces token voor beveiliging en een url om de organisatie te bereiken. Met deze gegevens kan er een “client” object aangemaakt worden, die de communicatie met de database regelt.

Via het zojuist aangemaakte “client” object kan er via kant-en-klare methodes van de “influxdb\_client” package weggeschreven worden. De “titels\_and\_sentiments” is een twee dimensionale array met alle titels en sentiments. Iedere titel wordt toegevoegd aan de “titels” measurement (tabel). Het correcte field of kolom wordt bepaald door het tweede element (die het sentiment voorstelt) van de “titel” array die op zijn beurt in de “titels\_and\_sentiments” array zit.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Voor het lezen wordt een query opgesteld. Alle rijen van de laatste 30 dagen (beperking van gratis versie) worden opgehaald uit de “titels” tabel. Voor iedere rij wordt gekeken uit welke kolom deze komt. Zo wordt een twee dimensionale array opgesteld met daarin drie arrays elk voor één sentiment. Alle titels worden toegoevoegd in de juiste array.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

Vervolgens kan in het “update.py” bestand de zojuist gemaakte twee dimensionale array bekeken worden. Zoals reeds vermeld worden de titels vanuit de cloud toch naar een lokaal file geschreven zodat de webpagina deze kan uitprinten met een iframe. Onderstaande code toont aan dat afhankelijk van in welke array de titel kwam het juiste file geopend wordt. Hierin wordt dan de titel geschreven via de ingebouwde “write” methode. Ook kunt u zien dat hier de letter ‘w’ wordt gebruikt, dit betekend dat het lokale bestand telkens overschreven wordt met alle titels uit de cloud.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

### Lokaal

Het lokaal wegschrijven van de titels is zeer gelijkaardig als bij lezen van de titels uit de cloud. Het grote verschil is dat het lokaal bijhouden van de titels enkel de nieuwe titels zal gebruiken. De “self.titels” zijn de nieuwe titels met hun sentiment, deze twee worden aan elkaar gezet via de “zip” methode van python en vervolgens naar een lijst omgezet. Hier wordt niet de letter ‘w’ gebruikt, maar wel de letter ‘a’ van append omdat enkel de nieuwe titels aan de file toegevoegd worden.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Het lezen van de lokale titels gebeurt automatisch bij het laden van de HTML-pagina via de iframe’s.

## Bluetooth Low Energy

### Arduino met OLED

Zoals eerder vermeld zal het OLED display het aantal titels per sentiment afbeelden. Enerzijds wordt bluetooth low energy gebruikt voor connectie tussen Arduino en Raspberry Pi. Anderzijds wordt I2C gebruikt voor communicatie tussen de Arduino en OLED display.

Afbeelding met tekst, elektronica

Automatisch gegenereerde beschrijving

De BLE connectie wordt geregeld via de “ArduinoBLE” library voor het Arduino gedeelte. Aan de Raspberry Pi kant wordt de “bluepy” package gebruikt. Bij het opstarten van de Arduino wordt als eerst een BLE-service aangemaakt via een uniek UUID. Met deze gegevens kan een ander bluetooth apparaat verbinding maken. Vervolgens wordt een characteristic gemaakt, dit beschrijft het formaat van de data-uitwisseling. Ook de characteristic krijgt een UUID en wordt aangemaakt via “BLEStringCharacteristic” omdat er tekst verstuurd zal worden.



De code gaat niet verder zolang de BLE connectie niet beschikbaar is via onderstaande if-structuur met een oneindige while loop.

Afbeelding met tekst

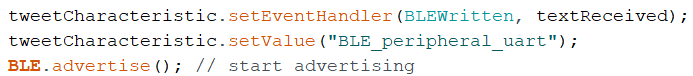
Automatisch gegenereerde beschrijving

Er wordt een lokale naam aangemaakt voor de service. Via “setAdvertisedService” wordt de naam peripheral gelinkt aan de UUID. Ook wordt de stringCharacteristic gelinkt aan de service. Tenslotte wordt de complete service toegevoegd.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Als laatste stappen wordt nog een eventhandler toegevoegd aan de characteristic via de “setEventHandler” methode. Als argumenten wordt meegegeven dat de “textReceived” methode wordt uitgevoerd wanneer er iets gewrite wordt op de characteristic. Na alle bovenstaande stappen kan de BLE-service eindelijk advertised worden naar de omgeving toe.



Om te kunnen verbinden met de Raspberry Pi is er een BLE adres nodig. Onderstaande code print het adres om dit te kunnen kopiëren.



In de oneindige void loop van de Arduio wordt enkel gekeken of er data ontvangen is via BLE. Dit gebeurt via de “BLE.poll” methode.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Indien er iets ontvangen werd, zal via de reeds besproken eventhandler een methode getriggerd worden. Onderstaande code toont aan wat er gebeurt na het ontvangen van data. De laatste value van het characteristic wordt opgevraagd. Deze data stelt de zojuist ontvangen string voor.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Om het OLED display aan te sturen wordt gebruik gemaakt van I2C. Dit gebeurt via de “Wire” en “Adafruit\_SSD1306” libraries. Om het OLED display in te stellen zijn enkele gegevens nodig waaronder de breedte en hoogte alsook het I2C adres dat via een hulp programma gevonden is. Het “display” object kan vervolgens aangemaakt worden via onderstaande code.

Afbeelding met tekst

Automatisch gegenereerde beschrijving



Opnieuw zal de code niet verder lopen zolang er geen correcte verbinding is.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Vervolgens wordt eenmalig het display gereset en worden de tekstinstellingen gekozen.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Nu kan de ingelezen string via BLE afgeprint worden op het OLED scherm via “display.println” na het triggeren van de eventhandler. Echter moet eerst het scherm gecleared worden, ook wordt de cursor in de linker bovenhoek geplaatst. Tenslotte moet er op het einde de methode “display.display()” opgeroepen worden vooraleer de tekst zichtbaar wordt.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

### BLE op Rapberry Pi

Telkens de “update.py” code wordt uitgevoerd, zal de functie van het “ble.py” bestand doorlopen worden.

Afbeelding met tekst, oranje

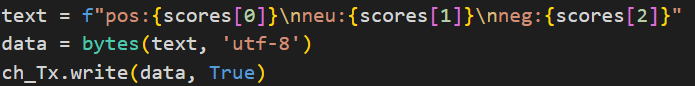
Automatisch gegenereerde beschrijving

Bij het ontwikkelen werd vastgesteld dat de BLE connectie niet altijd even stabiel was. Hierdoor is het belangrijk dat de code niet verder gaat vooraleer de connectie is opgesteld. Via “btle.Peripheral” kan er verbinding gemaakt worden naar het BLE adres van de Arduino. Vervolgens kan de correcte service van dat BLE adres via het unieke UUID opgehaald worden. Ook word het characteristic opgevraagd via zijn UUID. Er wordt gebruik gemaakt van een default delegate.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

De “scores” lijst bevat het aantal titels per sentiment. Via deze lijst wordt een string opgebouwd volgens onderstaande code. Om de data te versturen moet de string eerst omgevormd worden naar bytes volgens utf-8. Tenslotte kan er via de “write” methode data verstuurd worden.



## MQTT naar ESP32 met LED

Zoals reeds vermeld zal er een MQTT-communicatie zijn tussen Raspberry Pi en ESP32. Zo krijgt de ESP32 te weten hoeveel nieuwe titels er werden ingelezen bij het drukken van de knop op de webpagina. Vervolgens zal de ESP een aantal keer de led laten oplichten afhankelijk van hoeveel titels er werden ingelezen.

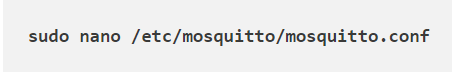
De Raspberry Pi functioneert als MQTT broker, maar zal ook als client publishen op een topic. De ESP32 zal als client subscriben op datzelfde topic. Er wordt gebruik gemaakt van Mosquitto om het MQTT-verkeer te leiden. Deze wordt volgens onderstaande afbeelding geïnstalleerd.



Onderstaand commando zorgt ervoor dat de Mosquitto service wordt uitgevoerd bij het opstarten van de Raspberry Pi.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Standaard na het installeren van Mosquitto zijn remote accesses niet toegelaten. Om dit toch verbinding te maken met de ESP32 wordt het mosquitto.conf bestand aangepast. 

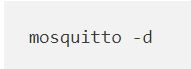
Onderstaande code werd in bovenstaand bestand toegevoegd.

Afbeelding met tekst, oranje, donker, sluiten

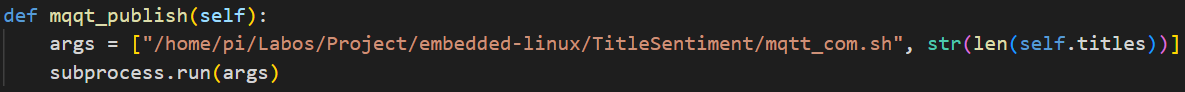
Automatisch gegenereerde beschrijving

Om de Raspberry Pi als client te laten functioneren moet onderstaande commando’s uitgevoerd worden.





In het “update.py” bestand wordt als laatst onderstaande methode uitgevoerd. Het “update.py” bestand werd vanuit het php-script uitgevoerd, terwijl het python bestand op zijn beurt een bash-script laat uitvoeren. Dit is mogelijk via de “run” methode van de “subprocess” package van python. Deze methode krijgt een lijst van argumenten namelijk: het te uitvoeren bestand en het aantal titels in string-formaat.



Het “mqtt\_com.sh” bestand dat vanuit de python code wordt uitgevoerd is zeer beperkt. Hier wordt enkel gepublished op een topic via de geïnstalleerde client. Het “mosquitto\_pub” commando wordt gebruikt met een meegegeven topic en het aantal titels dat hier “$1” voorstelt.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

De ESP32 wordt als client gebruikt via de bekende “PubSubClient” library. Ook wordt de “WiFi” library gebruikt. De wifi-client heeft onderstaande informatie nodig om te verbinden met het lokaal netwerk.

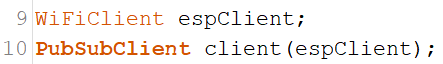


Om verbinding te maken met de Mosquitto broker wordt het ip-adres van de Raspberry Pi bijgehouden. Ook zal de led aan digitale pin 4 worden aangesloten.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Beide clients worden geinitialiseerd.



Vervolgens wordt de wifi-verbinding opgesteld via onderstaande methode. Via de “begin” methode van de “WiFi” library wordt een poging gemaakt, en zal dit blijven doen totdat er een verbinding is.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Eens er een wifi-verbinding is, kan de ESP32-client verbinding maken met de broker via onderstaande code. Met “setServer” wordt de client ingesteld met het correcte ip-adres van de Raspberry Pi of dus de Mosquitto broker en de standaard poort 1883. Ook wordt een callback of soort interrupt-methode ingesteld, die uitgevoerd wordt bij het ontvangen van data op een topic.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

De void loop is net zoals de BLE-connectie zeer beperkt. Hier wordt gekeken of de client nog steeds verbonden is met de broker, indien niet wordt de “reconnect” methode uitgevoerd. Vervolgens wordt via “client.loop()” gekeken of er nieuwe data gepublished is.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Onderstaande code zorgt voor een verbinding van client (ESP32) en de Mosquitto broker. Wederom zal de code niet verder lopen zolang er geen verbinding is. Hier wordt ook ingesteld dat de client moet subsriben op het topic genaamd ‘titels’. De verbinding wordt opgesteld via de “client.connect” methode.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Tenslotte nog de callback methode die telkens uitgevoerd wordt bij het ontvangen van data. Hier wordt bijgehouden op welk topic er een message is binnengekomen en welke message dit is, ook de lengte van deze message is bekend. Vervolgens kan gefilterd worden of dit het ‘titels’ topic is. Daarna wordt letter per letter of char per char de message opgesteld afhankelijk van de meegegeven lengte. Deze string waarde die het aantal nieuwe titels voorstelt (bv. 4) wordt omgevormd naar een int via de “toInt()” methode. Tenslotte wordt via een for-lus de led aan- en uitgeschakeld zoveel er nieuwe titels zijn.

Afbeelding met tekst

Automatisch gegenereerde beschrijving