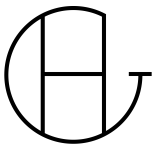


HoGent

Faculteit Bedrijf en Organisatie

The use of Internet of Things in Student Transportation

Glenn Goossens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Steven Vermeulen
Co-promotor:
Michiel Van Driessche

Instelling: —

Academiejaar: 2016-2017

Tweede examenperiode

Faculteit Bedrijf en Organisatie

The use of Internet of Things in Student Transportation

Glenn Goossens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Steven Vermeulen
Co-promotor:
Michiel Van Driessche

Instelling: —

Academiejaar: 2016-2017

Tweede examenperiode

Samenvatting

Het doel van dit onderzoek is IT meer integreren in het dagelijkse leven. Dit tracht men te bereiken door middel van Internet of Things toe te passen op alledaagse zaken en handelingen. In dit onderzoek zal Internet of Things gecombineerd worden met het studententransport bij uitstek, namelijk de fiets. Hierdoor wordt het dagelijkse leven van de student makkelijker, sneller en efficiënter gemaakt. Het gebruik van technologie op de fiets is de laatste jaren enorm gestegen, maar dit vooral in het competitief kader van fietsen. Zo kan je nu al electronisch schakelen, een gps-route volgen op de fiets of zelfs met een electrische fiets rond rijden. Deze bijdrage van technologie en fietsen gaat vooral om 'on the fly' gebruiken van een fiets in de buurt.

Hiervoor werd de basis gelegd bij het Uber concept dat heel erg populair is in Amerika. Dit concept houdt in dat je met behulp van een mobiele applicatie op je smartphone een rit kan aanvragen in plaats van te wachten op een taxi die misschien wel nooit komt. Op deze manier lijkt het interessant om een gelijkaardige mobiele applicatie te ontwikkelen voor de fiets die in verbinding staat met een klein computersysteem gemonteerd op de fiets. Een grondige uitleg kan je vinden op de website van Uber zelf, Uber (2016).

Het is niet de bedoeling dat je een passagier meeneemt op de fiets door deze mobiele applicatie, maar dat je je fiets zelf op een bepaalde manier verhuurt aan een willekeurig persoon. Stel dat je je bijvoorbeeld op de campus Schoonmeersen bevindt zonder fiets bij de hand en je hebt een beperkte tijd om de trein te halen in het station. Dan zal er hoogstwaarschijnlijk wel een fiets met het systeem op de campus Schoonmeersen staan die je kan gebruiken voor een korte rit naar het station. Aangezien Gent een heel resum aan fietsen heeft op elke straathoek, is dit de ideale kans om deze soms wel ongebruikte fietsen om te zetten in een freecycle systeem voornamelijk voor de student maar ook voor andere personen die er gebruik van willen maken.

Het systeem zal voornamelijk uit 2 grote componenten bestaan. Enerzijds de mobiele applicatie waar op de gebruiker een fiets rit kan aanvragen en anderzijds het systeem op de fiets die de fiets beveiligd en ontgrendeld wordt via de app. Laten we het eerst over het systeem op de fiets hebben. Kort omschreven kunnen we opmerken dat het gaat om een Raspberry Pi 3B waarop een zelfgeschreven Bluetooth Low Energy server draait waarop bluetooth devices request kunnen versturen en response ontvangen. Aan de hand van de requests en responses dat de server binnenkrijgt zal het bepaalde scripts op de Raspberry Pi 3B aanroepen en uitvoeren. Verder is de Raspberry Pi 3B ook nog voorzien van een alternator die rechtstreeks in contact staat met lichten die voorzien zijn op de fiets als eveneens de batterij voor de Raspberry Pi. Bij stilstand van de fiets en aanwezigheid van zonlicht zal de batterij kunnen opladen aan de hand van een klein zonnepanneel.

Anderzijds is er ook nog de mobiele applicatie. Deze applicatie is gebouwd voor Android devices die voorzien zijn van Bluetooth. Dit natuurlijk om de connectie met de bluetooth server op de Raspberry Pi 3B te kunnen maken. Op de mobiele applicatie zal je eerst en vooral kunnen aanmelden om vervolgens binnen de bluetooth reikwijdte een fiets op te sporen. Eens de fiets gelokaliseerd is, kan je ermee connecteren. Eens de connectie gemaakt is, kan je via een simpele knop een request sturen naar de fiets om deze vervolgens te ontgrendelen en te gebruiken. Op het einde van de rit kan je met een even simpele knop de fiets vergrendelen en achterlaten zonder zorgen.

Voorwoord

Het onderwerp van dit onderzoek komt vooral uit eigen ondervinding. Studeren aan de campus Schoonmeersen is de ideale plaats om verschillende transportmiddelen te gebruiken. Dicht bij het station als je met de trein komt. Makkelijk te bereiken met auto, en parking op school. Dus een fiets is niet echt een noodzaak voor mij als student. Tenzij je natuurlijk af en toe eens naar het centrum van Gent of de uitgaansbuurt wil bezoeken. Dicht bij het station zijn er wel trams en bussen maar door de drukte ervan, ga ik nog liever te voet. Een fiets kopen of huren bij mobiliteit Gent vind ik dan ook weer zonde omdat ik deze niet zo vaak zal gebruiken. En op die manier wordt mijn fiets dan een plaatsvuller zoals zovelen in de fietsenstallingen.

Door mijn grote interesse in fietsen en technologie leek het mij dan ook het ideale moment om een onderzoek waarbij beiden gecombineerd worden, te beginnen. Bovenal zet ik zo alles wat ik over de jaren heen geleerd heb om in de praktijk. Ik heb er voor gekozen om een mobiele applicatie voor Android op basis van Java, met daarnaast een Node.js bluetooth server met zelfgemaakte modules.

Inhoudsopgave

1. Inleiding

Zoals Morgan (2014) al simpel uitlegde in een artikel in 2014, kunnen we Internet of Things beschrijven als een concept dat de impact op ons leven en werk zal veranderen. Kort samengevat kan je Internet of Things benoemen als het concept om alledaagse objecten te connecteren met elkaar en informatie uit te wisselen. Zo kan bijvoorbeeld je koelkast gemonitord worden vanop je smartphone, waardoor je de temperatuur, inhoud en dergelijke altijd bij de hand hebt.

Dit zal natuurlijk een enorme impact hebben op het dagelijkse leven. Als je alles met elkaar gaat combineren in een soort netwerk van Internet of Things, dan zal dat naast positieve ook negatieve aspecten met zich meebrengen. Met alle handigheden dat het met zich meebrengt, zal er ook wel negatief gebruik gemaakt worden van alle sensors die aanwezig zijn. Alhoewel de impact van Internet of Things nog steeds het grootste vraagteken oplevert bij het concept zelf.

We integreren Internet of Things in dit onderzoek door connectiviteit toe te voegen op de fiets en deze te connecteren met een Android smartphone. Android is een mobiel besturingssysteem dat de basis is voor heel wat smartphones. Op het Android platform is het heel makkelijk om je eigen applicatie te maken en de wereld in te sturen. Op deze manier kunnen we onze eigen mobiele applicatie makkelijk verspreiden. Daarnaast heeft het alle componenten die nodig zijn om de gewenste functionaliteiten uit te bouwen.

De connectie op de fiets zelf zal tot stand gebracht worden door gebruik te maken van een Raspberry Pi 3B. Dit is een kleine computer die makkelijk programmeerbaar is om basis functionaliteiten uit te voeren in dit onderzoek. Door middel van een Bluetooth adapter kan je makkelijk een sterke Bluetooth connectie opstellen tussen de Raspberry Pi 3B en een Android smartphone. Door deze connectiviteit kan er informatie uitgewisseld worden tussen beide componenten in dit systeem.

1.1 Stand van zaken

We kunnen stellen dat de basis van ons onderzoek in alle te vormen componenten aanwezig zijn. Het is nu vooral de opdracht om elk component individueel uit te werken om ze vervolgens te verwerken in een mooi samenhangend geheel. Laten we de verschillende componenten opsplitsen in 2 delen, namelijk Android en Raspberry Pi 3B.

1.1.1 Android

Dit is een vrijwel makkelijk en kort uit te leggen component. Het mobiele besturingssysteem van Android is al een grondige basis voor de mobiele applicatie. Deze applicatie zal bestaan uit een eenvoudig login-systeem voor de gebruiker. De hoofdfunctionaliteit bevindt zich in het zoeken en connecteren met Raspberry Pi 3B's die op fietsen gemonteerd zijn. Eens de connectie tot stand gemaakt is, kan er informatie tussen beide uitgewisseld worden. Deze informatie kunnen we beschrijven als opdrachten die uitgevoerd worden op de Raspberry Pi 3B eens de gebruiker ze via de mobiele applicatie activeert.

1.1.2 Raspberry Pi 3B

De Raspberry Pi 3B is voor velen onbekend terrein. Daarom zal het misschien wel iets moeilijker zijn om dit component zo eenvoudig mogelijk uit te leggen. De Raspberry Pi 3B wordt omschreven als een kleine compacte computer die zo geprogrammeerd kan worden dat je er heel wat functionaliteiten aan kan toevoegen. Zoals het besturen van LED-lichten, informatie verkrijgen van sensors en dus ook connecteren met WiFi of Bluetooth. Dit component zal vooral bestaan uit een Node.js Bluetooth Low Energy server, een batterij module en de vergrendel module. De keuze voor dit specifieke model was snel gemaakt, aangezien dit model Bleutooth Low Energy support, Raspberry (2016).

Zoals eerder vermeld wordt er gebruik gemaakt van een Node.js server. Een server wordt meestal meteen geassocieerd met websites. Je stuurt een aanvraag in je webbrowser om een bepaalde website te bezoeken en de server geeft de inhoud van de website weer in de browser, Dickey (2017). In dit onderzoek loopt het proces gelijkaardig maar in plaats van het internet te gebruiken maken we gebruik van Bluetooth Low Energy en wordt er geen website weergegeven maar een opdracht uitgevoerd op de fiets.

Met deze opdracht wordt er bedoeld dat de vergrendel module aangesproken wordt en de

fiets ontgrendeld of vergrendeld wordt. Dit zal gebeuren aan de hand van een kleine servo motor, deze zal de schijfremmen van de fiets blokkeren of vrijmaken. De laatste module heeft betrekking op de batterij, want de Raspberry Pi 3B moet zijn energie van ergens halen. De batterij zal opgeladen worden door middel van een zonnepanneel bij stilstand en tijdens het fietsen door een alternator die tevens de verlichting van energie voorziet.

1.2 Probleemstelling en Onderzoeksvragen

De probleemstelling kunnen we omschrijven als het makkelijker en efficënter maken van het fietsverhuur van, in hoofdzaak, studenten. Hierbij kunnen we enkele concrete onderzoeksvragen stellen die elkaar aanvullen. Namelijk hoe kunnen we het fietsverhuur verbeteren in een studentenstad zoals bijvoorbeeld Gent? Waar liggen de knelpunten en hoe bereik en hoe overtuig je studenten om dit concept te gebruiken? Wat maakt het concept uniek en doorslaggevend om er gebruik van te maken? Hoe zal het concept opgesteld en moeten uitgevoerd worden in de realiteit?

1.3 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 3, worden alle componenten van het concept tot in detail toegelicht.

In Hoofdstuk 4, overlopen we alle voordelen dat dit systeem zal opleveren. Gaande van functionaliteit tot economische voordelen en verbeteringen voor zowel de gebruiker als de stad.

In Hoofdstuk 5, bekijken we de verbeterpunten. Aangezien dit een proof of concept is, is er dus ook altijd ruimte voor verbeteringen.

In Hoofdstuk 6, overlopen we de effectieve uitwerking van de verschillende componenten. Hierbij nemen we vooral de code onder de loep.

In Hoofdstuk 7, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

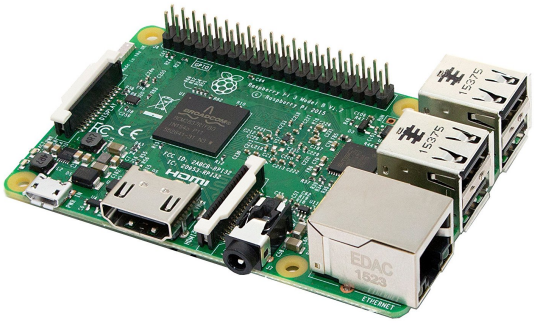
2. Methodologie

In dit hoofdstuk licht ik toe, hoe ik te werk ben gegaan in mijn onderzoek. Onderverdeelt in verschillende fasen, licht ik elke stap en ondervinding die ik tegenkwam toe. Daarnaast geef ik ook een woordje uitleg bij de keuzes en beslissingen die ik gemaakt heb. Zo kies ik altijd voor de best mogelijk oplossing binnen mijn bereik en mogelijkheden. De ideale oplossing heb ik in eerste instantie niet altijd kunnen bereiken. Dit meestal door te kort aan tijd en financiële fondsen. Een schriftelijke toenadering tot de ideale oplossing vind je terug in hoofdstuk 5. Echter wil ik wel mededelen dat ik ondanks niet de optimale oplossing gebruikt te hebben, wel de juiste antwoorden en uitwerkingen heb kunnen bereiken in mijn onderzoek.

2.1 Opzet Raspberry Pi 3B

De keuze voor een Raspberry Pi 3B was voor mij vrijwel eenvoudig. Ik had namelijk zelf de keuze tussen een Raspberry Pi 3B of een Arduino, andere mogelijkheden liet ik buiten schot. Maar aangezien ik meer ervaring heb de Raspberry Pi 3B en in eerste instantie deze meer mogelijkheden bood voor dit onderzoek. Om de Raspberry Pi 3B klaar te stomen voor dit onderzoek moest er echter wel nog heel wat gebeuren. Zo heb ik gekozen voor het Raspbian besturingssysteem. Omdat dit een veelgekozen besturingssysteem is dat zeker de nodige documentatie biedt indien problemen zouden opduiken. Verder zal er besproken worden hoe elk individueel component van de Raspberry Pi 3B werd opgesteld en welke middelen er werden gebruikt om dit mogelijk te maken.

Figuur 2.1: Raspberry Pi 3B



2.1.1 Batterij module

De zoektocht naar de juiste batterij module was in het begin vrijwel moeilijker dan gedacht. De standaard energietoevoer voor de Raspberry Pi 3B door middel van een micro USB oplaader bedraagt al 5 Volt aan 2.4 Ampére, terwijl het minimum voltage 3.7 bedraagt. Om dit draagbaar te maken had ik eerst gedacht aan de batterij van een oude GSM, maar omdat de praktische uitwerking en sterkte van deze batterijen niet optimaal waren, heb ik het idee gelaten voor wat het was. Een tweede gedachtengang dacht ik aan een Power Castle die gebruikt wordt om smartphones op te laden onderweg. Maar de kostprijs en mijn zin om zelf iets te creëren hield me wat tegen.

Het uiteindelijk idee was dan het bouwen van een eigen batterypack. Een 6-tal sterke AA-batterijen achter elkaar opstellen samen met een UBEC, Universal Battery Elemination Circuit dat dient als energieregelaat, zou een goedkope maar minder makkelijke oplossing zijn. Tijdens mijn zoektocht naar de juiste onderdelen om dit te bouwen, kwam ik echter een soortgelijke oplossing tegen met meerdere USB uit- en ingangen. De kleine opleg om het niet zelf te moeten maken, overtuigde mij om een kant-en-klare oplossing te gebruiken. Daarom heb ik dus gekozen voor een Raspberry Pi PowerPack dat 3.7V en 3.8 Ampére levert.

Verder zal het opladen van de Raspberry Pi PowerPack gebeuren aan de hand van een alternator en zonnepaneel. Deze onderdelen werden gehaald uit een opwindbare radio, die gebruikt kan worden tijdens het kamperen. De keuze hiervoor was heel erg simpel. De opwindbare radio bevat een alternator en zonnepaneel om een interne batterij op te laden en zorgt voor stroom voor de radio en een ingebouwde zaklamp. Aangezien de radio al USB-uitvoer poort had, was het makkelijk om deze om te leiden naar de Raspberry Pi PowerPack. Door het verwijderen van de interne batterij ging de opgeleverde energie rechtstreeks naar de Raspberry Pi PowerPack. Daarnaast gaat de toevoer van energie komende van de alternator ook naar de lichten die reeds op de fiets gemonteerd staan.

Figuur 2.2: Raspberry Pi PowerPack

2.1.2 Bluetooth Low Energy server

De keuze voor Bluetooth Low Energy is vanzelfsprekend voor dit onderzoek. Bluetooth Low Energy zegt het in de naam al zelf, het heeft een veel kleiner verbruik en is dus ideaal voor projecten waar de energievoorraad beperkt is. Eerst heb ik geprobeerd om de standaard ingebouwde bluetooth op de Raspberry Pi 3B in te stellen als Bluetooth Low Energy maar dit werd niet ondersteund. Een Bluetooth 4.0 USB adapter was dus de enige oplossing.

Om informatie-uitwisseling tussen de Raspberry Pi 3B en de Android mobiele applicatie te verkrijgen, was ook de standaard ingebouwde bluetooth op de Raspberry Pi 3B niet voldoende. Na wat opzoekwerk kwam ik de Node.js 'bleno' module tegen op het internet. Node.js servers worden gebruikt als backbone voor heel wat fullstack websites. Node.js maakt gebruik van JavaScript om een webserver te maken aan de hand van modules. In dit onderzoek maken we gebruik van de 'bleno' module. Deze module zorgt ervoor dat alle communicatie van requests en responses gebruik maken van Bluetooth Low Energy in plaats van het internet.

Om de eerste communicatietest uit te voeren heb ik het echo voorbeeld uit de bleno documentatie gebruikt en geïmplementeerd op de Raspberry Pi 3B. Als communicatietest vanuit het Android mobiele gedeelte, heb ik de mobiele applicatie 'nRF Connect' gedownload op mijn persoonlijke smartphone. De eerste connectie kwam niet tot stand ondanks dat de server aangaf nochtans operationeel te zijn. De mobiele applicatie 'nRF Connect' kon geen Bluetooth Low Energy apparaten opmerken in de nabije omgeving. Na wat opzoekingswerk bleek dat de standaard ingebouwde Bluetooth op de Raspberry Pi 3B de Bluetooth Low Energy USB adapter verwierp. De oplossing hield in dat de standaard ingebouwde Bluetooth uitgeschakeld moest worden en de Bluetooth Low Energy USB adapter ingesteld moest worden als huidige Bluetooth connector.

Na de eerste test werd het echo voorbeeld uit de bleno documentatie vervormt naar een persoonlijke toepassing voor dit onderzoek. Hierbij wordt er geluisterd naar inkomend bericht met de inhoud 'open', indien dit bericht binnenkomt, wordt er een intern script aangeroepen dat de servo motor bestuurt en een succes response gestuurd naar de Android mobiele applicatie.

2.1.3 Vergrendel module

Deze module bestaat uit een simpele servomotor namelijk de SG90. Deze kleine servomotor heeft een hoge output ondanks zijn kleine gewicht. Met een draaivermogen van 180 graden kan het dus makkelijk bestuurd worden in deze vergrendel module. Dat deze servomotor als basis voor het vergrendelen zou gebruikt worden, was al snel beslist. Maar wat de uiteindelijke uitwerking van de vergrendel module zou omvatten, had enig praktisch denkwerk nodig. Het eerste idee was de motor inzetten om een ringslot te besturen. Van deze sloten wordt er reeds gretig gebruik gemaakt bij stadsfietsen. Maar aangezien de servomotor maar een draaivermogen heeft van 180 graden, werd dit idee al snel van de baan geschoven.

Het volgende idee was het concept van een automatische deur zodanig herwerken dat het op de fiets zou toegepast kunnen worden. De servomotor werd even aan de kant geschoven. Het systeem van een automatisch deur bestaat simplistisch geschetst uit een pin die uitsteekt als het slot gesloten is en zich terugtrekt als het slot open gaat. Op deze manier zouden we een pin door het wiel kunnen schuiven om zo het wiel te blokkeren als het slot gesloten is. Maar dit is niet zo praktisch aangezien de pin te gemakkelijk zou breken als ze niet stevig genoeg is en te ver aan de buitenkant zou uitsteken als het slot niet gesloten is.

Het uiteindelijke idee bestaat uit de eerder vermelde servomotor die met zijn 180 graden draaivermogen de remmen van fiets zal besturen. Door gebruik te maken van schijfremmen. Heb je niet enkel een betrouwbaarder en sterker remvermogen, maar kan je deze ook makkelijk klem zetten. Waardoor de vergrendel module dus zal bestaan uit een servomotor dat door de draaibeweging een veer tegentrekt en dus de schijfrem vast klemt als het slot gesloten is. Als het slot open is, dan staat de servomotor in een neutrale positie en trekt de veer het schijfremblokkerende onderdeel weg van de schijfrem.

Figuur 2.3: SG90 servo motor

2.2 Android mobiele applicatie

Als basis voor de mobiele applicatie heb ik gekozen voor Android. Naast Android had ik ook nog de keuze om een app te maken voor Windows of iOS. Toch heb ik voor android gekozen omdat deze gebaseerd is op Java , Java (2017). Java is een programmeertaal en computerplatform dat heel wat toepassingen heeft. Hierdoor is Android eenvoudig overdraagbaar naar andere systemen wat iOS niet heeft en Windows nog niet heeft. Daarnaast kan je Android mobiele applicaties programmeren in Android studio. Android studio is een sublieme ontwikkelaarsomgeving gemaakt door JetBrains ‚JetBrains (2017). Door Android studio kan je makkelijk en snel mobiele applicaties ontwikkelen door zijn Gradle-based build systeem Gradle (2017), live-layout en vele support.

Daarnaast is een Android mobiele applicatie makkelijk te verdelen via de Google play store. En de belangrijkste reden is de lage instapkosten. Bij iOS moet je namelijk een Mac en een jaarlijks abbonement van €100 hebben om je mobiele applicatie de wereld in te sturen. Terwijl je bij Android een eenmalige betaling van €25 moet neerleggen om al je mobiele applicaties de wereld in te sturen. De keuze van Windows lag ver buiten schot want Windows heeft namelijk maar een heel klein marktaandeel op de mobiele besturingssystemenmarkt.

2.2.1 Bluetooth Low Energy

De keuze voor Bluetooth Low Energy is hier vanzelfsprekend aangezien we op de Rapsberry Pi 3B ook gebruik maken van Bleutooth Low Energy als connectiepunt. Bleutooth Low Energy is op Android beschikbaar sinds Android 4.3 (API level 18). Hierdoor kunnen mobiele applicaties apparaten ontdekken, informatie uitwisselen of bepaalde diensten opvragen. Bleutooth Low Energy wordt veelal gebruikt om kleine hoeveelheden informatie uit te wisselen of te communiceren met kleine sensors in de nabijheid. Het grote voordeel van Bluetooth Low Energy is , zoals de naam het al zegt, de veel lagere consumptie van energie. Dit gaat ten goede van de batterij van zowel de apparaten waarmee het communiceert als de batterij van de smartphone.

Het uitwerken van de backend met Bleutooth Low Energy verliep vlotter dan gedacht. Dit omdat de documentatie van Android alles subliem uitlegt en er enkele tutorials zijn die dicht aanleunen bij wat ik wil bereiken. Een kant en klare oplossing bestond er niet maar wel enkele goede handleiding en best practices die leiden tot wat het nu is. Even kort opgesomd, hield het opstellen van de mobiele applicatie voor Bleutooth Low Energy in dat de juiste toestemmingen aangevraagd moeten worden. Om daarna met het gebruik van de BleutoothAdapter binnen Android de juiste operaties uit te voeren.

2.2.2 Functionaliteiten

De hoofdfunctionaliteiten van de mobiele applicatie bestaan vooral uit operaties om de connectie en communicatie met de Rapsberry Pi 3B tot stand te brengen door middel van Bleutooth Low Energy. De belangrijkste functionaliteit dat deze mobiele applicatie meedraagt, is het verzenden van kleine boodschappen naar de Raspberry Pi 3B door middel van Bluetooth Low Enerqy. In dit prototype zijn de boodschappen heel erg minimalistisch voorgesteld met woorden die de operatie omschrijven zoals open en close. Door deze boodschappen weet de Raspberry Pi 3B welke operaties hijzelf moet uitvoeren. Daarnaast is er ook nog de zoekfunctie. In deze functie kan je met de mobiele applicaties Bluetooth Low energy apparaten zoals de Raspberry Pi 3B in dit voorbeeld zoeken. Nadat je een Raspberry Pi 3B hebt gevonden, kan je hiermee connectie maken. Door de connectie vervalt de mogelijkheid van connectie met dit bepaald apparaat bij andere gebruikers en kan je zonder problemen zelf kiezen welke operatie je wil uitvoeren.

Daarnaast had ik nog het idee om enkele kleine extra interessante functionaliteiten toe te voegen. Zoals de mogelijkheid om in te loggen met credentials en het toevoegen van een kleine databank. Deze beide functionaliteiten zullen steunen op FireBase , Firebase (2017). Firebase is een platform voor web en mobiele applicaties waarbij ontwikkelaars functionaliteiten kunnen toepassen naar hun eigen noden. Hierdoor kan je makkelijk een login systeem en database toevoegen zonder al te veel rekening te houden met extra's om deze werkende te krijgen. Hierdoor kan je je beter focussen op de hoofdfunctionaliteiten van je applicaties. Ik heb voor dit systeem gekozen omdat ik er al ervaring mee had en er zich toen geen problemen voordeden. Deze goede ervaring zorgde dan ook voor een logische beslissing en oplossing van deze functionaliteiten.

3. Componenten

In dit hoofdstuk worden alle componenten nog eens overlopen en in detail uitgelegd wat hun specificaties zijn. Dit overzicht is bedoeld als een technische opsomming van alle componenten die deel uitmaken van het onderzoek. Het gaat hier vooral om de functionaliteiten die toedragen tot de werking van het geheel. In dit onderzoek werd er niet gekeken naar performantie of overdrachtsnelheden omdat deze geen betrekking hebben op de werking van het volledige systeem.

Allereerst overlopen we de technische aspecten van de Raspberry Pi 3B en zijn toebehoren. Deze kunnen we onderverdelen in de batterij module, Bluetooth Low Energy server en vergrendel module. Daarna overlopen we de Android mobiele applicatie. Specifiek wordt er meer verteld over de techniek die gebruikt werd om de Bluetooth Low Energy connectie te maken en de verschillende functionaliteiten die de Android mobiele applicatie te bieden heeft.

3.1 Raspberry Pi 3B

De Raspberry Pi 3B zal draaien op het Raspbian besturingssysteem. Raspbian is een gratis besturingssysteem dat gebaseerd op Debian en geoptimaliseerd is voor de Raspberry Pi 3B hardware. Het besturingssysteem bestaat uit een selectie van basis programma's en tools die het mogelijk maken om de Raspberry Pi 3B operationeel te maken. Naast de basis functionaliteiten om de Raspberry Pi 3B operationeel te maken, biedt Raspbian ook meer dan 35.000 duizend packages en voor gecompileerde software gebundeld in een compact formaat. Raspbian valt niet onder de Raspberry Pi Foundation maar is gecreëerd door ontwikkelaars die een voorliefde hebben voor Raspberry Pi en Debian. Raspbian (g.d.)

Naast de voor gecompileerde software en packages zal het gebruik van de GPIO op Raspberry Pi 3B in dit onderzoek een groot aandeel hebben. GPIO is de afkorting van General Purpose Input/Output, deze pinnen zijn de fysieke interface tussen de Raspberry Pi 3B en de externe componenten. Naast de 4 standaard geleverde USB poorten op de Raspberry Pi 3B is het mogelijk om een input/output circuit op te stellen via de GPIO. Zo zijn er pinnen die zorgen voor stroomtoevoer of aarding en andere die luisteren naar input of verzenden van output. Op deze manier zullen we de kleine servo motor besturen om de vergrendelmodule operationeel te maken binnen de context.

Verder werd er ook nog gebruik gemaakt van enkele externe libraries om de Bluetooth Low Energy werkende te krijgen op de Raspberry Pi 3B. Zo werd als eerste de standaard Bluetooth service uitgeschakeld en via hciconfig, dit commando configureert Bluetooth devices op Linux gebaseerde besturingssystemen, de Bluetooth Low Energy adapter opgezet. Daarnaast werd de laatste nieuwe stabiele versie van Node.js geïnstalleerd. Ook de libraries voor Bluetooth development mogelijk te maken, werden niet vergeten.

3.1.1 Batterij module

Zoals eerder vermeld bestaat de batterij module uit een Raspberry Pi PowerPack gecombineerd met de energietoevoer vanuit een opwindbare radio. De Raspberry Pi PowerPack bestaat uit een 3.8 Ampére lithium batterij, dat ervoor zorgt dat deze in elke mogelijk manier opgesteld kan worden en niet onderhevig is aan de positie van vloeistoffen in de batterij. De batterij heeft een maximale ontlaadstroom van 1.8 Ampére en een onbelaste uitgangsspanning van 5.1 Volt. De standaard laadstroom/spanning bedraagt 1 Ampére/ 5 Volt en heeft een Cut-off spanning bij het volledig opladen tussen 4.18 en 4.2 Volt. Verder is de batterij nog voorzien van 2 USB Type-A uitgangpoorten en 1 miscro USB inganspoort. Deze batterij is speciaal ontworpen voor de Raspberry Pi 3B en heeft dan ook de nodige connecties om aan de Raspberry Pi 3B te bevestigen. Tevens heeft een volle batterij een levensduur van 9 à 10 uur. raspberrywiki (2016)

De stroomtoevoor zal voorzien worden door een alternator komende uit de opwindbare radio. Deze alternator heeft een werkspanning van 2.7 Volt tot 4.2 Volt en zal zowel de batterij van energie voorzien alsook de lichten van de fiets. Daarnaast is er ook het zonnepaneel dat de batterij van energie zal voorzien tijdens het stilstaan van de fiets indien deze in de nabijheid van licht is gepositioneerd. Het zonnepaneel heeft een werkspanning van maximaal 4 Volt.

3.1.2 Bluetooth Low Energy server

Node.js is de absolute topper in real-time web applicaties door gebruik te maken van sterke websockets. Node.js gebruikt 1 thread om events te behandelen terwijl er traditioneel een thread per event werd aangemaakt. Daardoor is Node.js sneller in het handelen van requests en responses, aangezien er geen wachttijd op een andere thread voorzien is. Node.js is licht en snel, en dus de perfecte basis om modules te schrijven voor de Raspberry Pi 3B die tevens kijkt naar zijn energieverbruik. Door het gebruik van modules op Node.js kan je heel wat makkelijker een server naar eigen wens opstellen. Dickey (2017)

In dit onderzoek werd er gebruik gemaakt van de bleno module. Deze module is gemaakt om Bluetooth Low Energy randapperatuur te implementeren. De module wacht tot er een externe connectie kan gemaakt worden en wacht vervolgens op communicatie met het externe device. In de module kan je zelf je service opstellen die verschillende karakteristieken bevat. Zo is er in dit onderzoek een service gemaakt die de karakteristiek heeft om te luisteren naar input en output. Deze input en output worden verwerkt binnen de karakteristieken zelf en handelen afhankelijk van de meegegeven informatie. Zo kan er binnen een bepaalde karakteristiek een script gestart worden dat de vergrendelmodule bestuurt. Daarnaast kan de bleno module zo ingesteld worden dat het maar 1 connectie aanvaardt. Als er connectie wordt gemaakt dan verdwijnt de connectie mogelijkheid van andere apparaten. Daardoor kan je geen rijdende fiets op slot doen vanop een ander apparaat.

3.1.3 Vergrendelmodule

Het hoofdonderdeel van de vergrendel module bestaat uit de SG90 servomotor. Met een gewicht van 9g en kubistische afmetingen van 22.2mm op 11.8mm op 31mm, is dit een klein lichtgewicht. Ondanks dat deze servomotor zo klein is in omvang en gewicht, heeft het een sterke output van 1.8kg * cm. Het draaivermogen bedraagt 180 graden met een snelheid van 0.1 seconde per 60 graden. Met een werkvoltage van 4.8V is het dus conform te gebruiken op de Raspberry Pi 3B. De rotatie van de arm wordt via de pulsbreedtemodulatie doorgegeven aan de hand van milliseconden. Elke arbeidscyclus van de servomotor bestaat uit een puls van 50 Hertz van een aantal milliseconden. Door de pulsbreedtemodulatie berekent de servomotor aan de hand van het aantal miliseconden de richting van de draaibeweging. Micropik (2016)

3.2 Android Mobiele Applicatie

Android is een besturingssysteem voor mobiele telefoons, tablets en meer. Het is gebaseerd op een Linuxkernel en de Java programmeertaal. Android is het meest verkochte besturingssysteem op mobiele telefoons en dus het ideale platform om deze mobiele applicatie de wereld in te sturen. Android is ontwikkeld door Android inc., een bedrijf dat in 2005 door Google werd overgenomen. Orgineel was het idee achter Android het ontwikkelen van een besturingssysteem voor camera's met ingebouwde cloud-mogelijkheden. Maar Android wordt nu voornamelijk gebruikt voor mobiele telefoons, tablets en ook camera's. Op 5 November 2007 kondigde Google Android aan als besturingssysteem voor mobiele telefoons en tablets. In samenwerking met heel wat telecommmunicatiebedrijven werd Android een open standaard voor de mobiele telefonie.

De eerste Android versie werd op 23 september 2008 uitgebracht met als codenaam "Angle Cake". Ondertussen zitten we al 8 jaar en 25 versies verder. Momenteel is Android 7.1 met als codenaam "Nougat"de laatste nieuwe versie die uitgebracht werd op 4 oktober 2016. Maar wat voor ons van belang is, is de oplevering van Android 4.3. Sinds Android 4.3 heeft Android een ingebouwd platform dat Bleutooth Low Energy ondersteunt. Hierbij speelt Bleutooth Low Energy een centralere rol waarbij de mobiele applicaties apparaten kan ontdekken, informatie uitwisselen en diensten opvragen.

3.2.1 Bleutooth Low Energy

In deze sectie leg ik zo kort, bondig en zo volledig mogelijk de hoofdconcepten en termen van Bleutooth Low Energy uit binnen Android, alsook de verschillende rollen en verantwoordelijkheden. Hierbij bespreken de Generic Attribute Profile, het Attribute Protocol, een Charasteristic, een Descriptor en een Service. Deze termen hebben allemaal betrekking tot Bleutooth Low Energy en zullen hiervoor dus ook specifiek besproken worden. Bij de rollen en verantwoordelijkheden van Bleutooth Low Energy binnen Android kan men heel kort het verschil tussen een GATT client of server en het verschil tussen een centrale of peripheral uitleggen.

Generic Attribute Profile of afgekort GATT, is algemene specificatie van een profiel voor het verzenden en ontvangen van kleine hoeveelheden informatie, die ook wel attributes genoemd worden, over de Bleutooth Low Energy connectie. Alle profielen voor Bleutooth Low Energy zijn gebaseerd op GATT, deze profielen worden op de website van Bluetooth SIG gedefinieerd, Bleutooth (2017). Deze profielen bevatten specificaties over hoe een bepaald apparaat werkt in een bepaalde applicatie. Een apparaat kan meer dan een profiel definiëren, zo kan bijvoorbeeld een apparaat zowel een hartslagmeter als een batterij-indicator.

GATT is gebouwd bovenop ATT ofwel het Attribute Protocol. ATT is zo geoptimatiseerd dat het kan gebruikt worden voor Bleutooth Low Energy apparaten, dit door gebruik te maken van zo weinig mogelijk geheugen. Elk attribute of kleine hoeveeldheid informatie dat verstuurt wordt tussen verschillende Bleutooth Low Energy componenten, is uniek gedefinieerd door een UUID of Universally Unique Indentifier. Hierdoor kan elk deel informatie uniek geïdentificeerd worden. De attributes die door ATT getransporteerd worden over Bleutooth Low Energy , worden geformatteerd als Charasteristics en Services.

Een Characteristic is een enige waarde met enkele Descriptors. Deze Descriptors beschrijven de enkele waarde van de Characteristic. Een Characteristic kan aanzien worden als een type dat een bepaalde waarde definieert. Hierbij speelt de Descriptor een belangrijke rol. De Descriptor definieert de attributen die de waarde van de Characteristic beschrijven. De Descriptor beschrijft eigenlijk de Characteristic in lekentaal aan de hand van restricties of bepaalde meetwaarden. Een Service is op zijn beurt een collectie van verschillende Characteristics. Laten we dit even verduidelijken aan de hand van een voorbeeld. Laten we een hartslagmeter als voorbeeld nemen. Hierbij heb je een Service dat Characteristics bevat. De Service heeft zo de Characteristic ’hartslag meten’ waarbij de hartslag gemeten wordt. Binnen deze "Characteristic"beschrijft de Descriptor de voorwaarde waaraan een hartslag moet voldoen. Zo zal dat bijvoorbeeld al zeker een getal moeten zijn tussen de 0 en 220 zijn. Naast de Characteristic ’hartslag meten kunnen er nog andere Characteristics gedefinieerd worden binnen de Service. Elke Characteristic heeft dan zijn eigen algemene beschrijving van voorwaarde opgelegd door de Descriptor.

Als laatste bespreken we kort de rollen en verantwoordelijkheiden van Bleutooth Low Energy binnen Android. De Bleutooth Low Energy connectie heeft 2 mogelijke rollen. Zo heb je de centrale rol waarbij het apparaat zoekt naar mogelijke connecties. Terwijl de periferische rol zijn connectie open stelt voor andere apparaten. Eens de connectie tot stand is gekomen, kunnen we spreken over verantwoordelijkheiden. Deze verantwoordelijkheiden worden opgesplitst in client of server. Dit bepaalt de manier waarop apparaten met elkaar communiceren eens de connectie er is. Als verduidelijkend voorbeeld nemen we dit systeem. Hierbij hebben we 2 apparaten namelijk de Raspberry Pi 3B en de mobiele applicatie. Waarbij de Raspberry Pi 3B de centrale rol vertolkt en de mobiele applicatie de periferische rol heeft. Beide kunnen zowel de server als de client rol vertolken als de connectie tussen beiden gemaakt is. Zo zal je initieel de Raspberry Pi 3B willen aanspreken om de fiets te ontgrendelen. Hierbij is de mobiele applicatie de server die informatie doorspeelt aan de Raspberry Pi 3B, die op dat moment de client speelt. De Raspberry Pi 3B stuurt een confirmerend bericht terug naar de mobiele applicatie waarbij de rollen omgedraaid worden.

Android, 2017

3.2.2 Functionaliteiten

Zoals eerder vermeld hoort de communicatie en het uitwisselen van informatie door middel van Bleutooth Low Energy tot een van de hoofdfunctionaliteiten van deze mobiele applicatie. Maar omdat we in de vorige sectie hier al diep genoeg op ingegaan zijn, bespreek ik hier enkel de andere functionaliteiten. Deze andere functionaliteiten bestaan uit het aanmelden als gebruiker en het bijhouden van gegevens in een database. Dit zal allemaal gebeuren door Firebase toe te voegen aan de mobiele applicatie. Firebase is een platform voor web en mobiele applicaties waarbij je makkelijk functionaliteiten zoals een login-systeem of een real-time database kan toevoegen. Daarnaast is er ook nog de mogelijkheid om eenvoudig pushberichten naar de mobiele applicatie te sturen.

Hiervoor moet de Firebase SDK toegevoegd worden aan het Android project. SDK staat voor software development kit en bestaat uit een verzameling van hulpmiddelen die handig zijn bij het ontwikkelen van computerprogramma's. Door de Firebase SDK toe te voegen aan het project, creëer je extra mogelijkheden voor je mobiele applicatie. Zo kan je firebase-auth toevoegen waardoor makkelijk een authenticatie systeem kan toevoegen. Firebase zorgt dan voor heel wat achtergrondprocessen zodat je enkel de focus nog moet leggen op het maken van een goede login activiteit. Naast authentication heb je ook nog firebase-database en firebase-messaging. Deze voegen respectievelijk een databank en pushberichten toe aan je mobiele applicatie. De databank zorgt voor het opslaan van gegevens in de cloud terwijl je met pushberichten notificaties kan sturen naar de mobiele applicatie.

4. Voordelen

In dit hoofdstuk overlopen we, zoals de titel het al aanduidt, de voordelen van dit systeem. De hoofdreden bij het creëren van iets nieuws, is natuurlijk altijd verbetering. Maar dit is niet altijd het geval. In dit werkstuk wil ik een verbetering maken op het fietsgebruik in Gent, met als hoofddoelgroep de studenten. Deze doelgroep werd specifiek gekozen omdat zij fanatieke gebruikers zijn van de fiets als transportiemiddel in Gent. En omdat zij na hun studies hun transportmiddel achter laten in Gent, iets wat dit systeem kan opvangen. Ondanks dat deze doelgroep specifiek werd gekozen, wil dit niet zeggen dat andere doelgroepen geen gebruik kunnen maken van dit systeem.

Verder bespreken we de voornaamste voordelen dat dit systeem met zich meebrengt. Met name de verbeteringen die het grootste verschil aanduiden ten opzichte van de huidige situatie. Hier bekijken we eerst en vooral de vergelijking met andere vervoersmiddelen die ter beschikking staan in het algemeen. Daarna bespreken we de verbeteringen op de fiets zelf. Alsook de voordelen dat de technologie met zich mee brengt en hoe deze kan bijdragen voor zowel de gebruiker als de verdeler.

4.1 Vergelijking van de transportmogelijkheden

Dit systeeem is bedoeld als transportmiddel voor de student. Laten we een concreet voorbeeld stellen zodat we een makkelijk referentiepunt hebben. Vooral omdat dit systeem bedoeld is voor de student die zich moet verplaatsen vanuit zijn of haar kot naar de campus, of van het station naar de campus, of van campus naar campus. Daarom stellen we dit als ons referentiepunt op. Als student in Gent heb je maar enkele oplossingen om zich op deze manier te transporteren. Zo kan iedereen te voet gaan, met de tram of bus , met de fiets en voor sommige kan deze korte afstand zelfs met de auto afgelegd worden.

De aankoop van een Buzzypas voor 12 maanden kost je als student €204. Hiermee kan je als student onbeperkt reizen op alle geregeld vervoer van DeLijn, 2017. Dit is voor een student een grote jaarlijkse kost maar tevens ook een goede investering. Want deze Buzzypas kan gebruikt worden zowel binnen als buiten Gent doorheen het volledige jaar. Het grootste minpunt van dit vervoersmiddel is de stiptheid en wachttijd van de bussen of trams. Want zij maken deel uit van het grote verkeer en kunnen dus opgehouden worden door andere voertuigen of werkzaamheden. De student wenst natuurlijk ook op tijd op zijn eindbestemming te geraken en dan is dit vervoersmiddel niet altijd de ideale oplossing. De drukte en afhankelijkheid van anderen speelt hier zeker niet in het voordeel.

Met de fiets of te voet is voor een student de makkelijkste, goedkoopste, sportiefste en milieuvriendelijkste oplossing. Je bent tevens minder afhankelijk voor anderen in het verkeer en kan je dus in de stad makkelijk van punt A naar punt B verplaatsen. Het enige nadeel van dit transportmiddel zijn de mogelijke weersomstandigheden. Het is natuurlijk niet zo prettig om volledig doorregend aan te komen op je bestemming. Daarvoor is de bus of tram dan een beter alternatief om droog en warm van punt A naar punt B te gaan. Maar toch moet het gebruik van de fiets meer gepromoot worden, zeker in de stad. Het is veel gezonder en sportiever om je met de fiets te verplaatsen. Als iedereen zich meer met de fiets of te voet zou verplaatsen, dan zouden de schadelijke stoffen in de lucht verminderen door het vermindert voorkomen van uitlaatgassen. Wat zou toedragen tot een gezondere en aangenamere stad om te vertoefen. Waarom dan toch de fiets in plaats van te voet? Time is money zoals men zegt en met de fiets ben je gemiddeld 3 maal zo snel op je eindbestemming.

Voor sommige studenten wordt vaak een korte afstand afgelegd met de auto. We kunnen duidelijk stellen dat dit transportmiddel niet de meest geschikte methode is. Zo kunnen we al snel heel wat nadelen opsommen die meer opwegen dan de voordelen. Het grote voordeel dat de auto met zich meebrengt, is het gemak. Al moeten we het woord gemak wel specificeren in dit geval, want niet alles zal met gemak gaan. Je hebt het gemak om je persoonlijk vervoer te hebben. Je hebt het gemak om bij slechte weersomstandigheden in een warme, droge auto te zitten. Maar verder zijn we al door de voornaamste voordelen van een auto heen, als het gaat om transport in de binnenstad. De grote nadelen van een auto in de binnenstad is dan wel de parking. Niet alleen het vinden maar ook de prijs die je voor een parking betaalt zijn niet al te min. Daarnaast kan je in de stad niet makkelijk doorrijden, file stapelt zich dus al makkelijk op en zo ben je met de auto langer onderweg dan elk ander transportmiddel in de stad. Als je dan nog eens het nieuwe circulatieplan in Gent in acht moet nemen, dan zijn de voordelen van een auto volledig weg als het gaat om gemak en snelheid.

Maar waarom is dit systeem beter dan de Blue-bike of de gele fietsen van student en mobiliteit? We kunnen dit heel simpel uitleggen want dit systeem is in grote lijnen eigenlijk een samensmelting van deze 2 reeds bestaande oplossingen. Blue-bike is een systeem waarbij je aan meer dan 50 NMBS stations, De Lijn haltes en Park & Ride's een fiets kan huren en gebruiken tot je terug op een van de Blue-bike stations bent. Het kost je €12 per jaar om lid te worden en maximaal €3.15 per dag om een fiets te huren bij Blue-bike, 2017. Iedereen kan dit systeem gebruiken zolang je maar lid bent bij Blue-bike. Bij student en mobiliteit kan je enkel maar een fiets huren als je een student bent aan een Gentse hogeschool of universiteit. Zo'n gele fiets huren voor 12 maand kost je als student €110 waarvan er €60 waarborg is StudentEnMobiliteit, 2017. Als student is dit dus een goedkoper alternatief dan de Blue-bike of een Buzzypas.

In principe willen we met dit systeem een combinatie doen van de gele fiets van student en mobiliteit en de Blue-bike. Het principe dat een student goedkoop een fiets kan huren zoals bij de gele fietsen van student en mobiliteit. En het principe dat er altijd wel ergens een fiets klaarstaat om direct te huren zoals bij Blue-bike, wil ik hier combineren. Met dit nieuwe systeem wordt het gemak van de Blue-bike en de goedkope verhuurprijzen voor de student het nieuwe ideale transportmiddel. Tevens is, zoals eerder vermeld, de fiets een sportief en milieuvriendelijk transportmiddel dat zowel een voordeel is voor de stad en hun natuur als voor de fitheid van de gebruiker.

4.2 Verbeterde fiets

Naast het systeem is ook het hoofdcomponent van dit alles een grote verbetering ten opzichte van de mogelijkheden waarover men nu beschikt. Dit alles kwam tot stand door te kijken naar wat er nu is en wat er beter kan of wat er zelfs niet goed was. Een simpele analyse hiervan leidde tot een verbeterde fiets ten opzichte van de gele fiets van student en mobiliteit en de Blue-bike. Zoals eerder vermeld was dit weer een combinatie van het beste uit beide systemen. Beide systemen hebben al een opvallende fietskleur waardoor ze opvallen. Een opvallend fietskleur is namelijk van groot belang voor de veiligheid en zichtbaarheid in verkeer. De fietser is en blijft een zwakke weggebruiker, zeker in een bruisende stad zoals Gent. Daar moet dus zeker iets aan gedaan worden, daarom koos ik voor een fluogroen kader dat zowel opvalt overdag als in het donker.

Naast het kleur zijn er ook andere verbeterpunten op de fiets aangebracht. Om bij het thema zichtbaarheid in het verkeer te blijven, is de dynamo van het oplaadsysteem verbonden met de voorlichten en achterlichten. Niets speciaals ten opzichte van een gewone stadsfiets waar dit al een veelgebruikt systeem is. Maar hier wil ik even duidelijk maken dat de dynamo de batterij van het systeem en de lichten van stroom voorziet, waardoor de lichten geen gebruik maken van de batterij. Hierdoor gaat alle energievoorziening van de batterij naar het vergrendelsysteem. Wat er dan wel verbeterd is ten opzichte van de andere fietsen, zijn de remmen. Uit eigen ervaring kan ik zeggen dat het gemak, snelheid, onderhoud en remvermogen van schijfremmen heel wat beter is ten opzichte van gewone V-brakes. Zoals Huang, 2016 in een artikel uitlegt, kan je zien dat schijfremmen een sterker remvermogen hebben. Terwijl de meeste studenten in Gent op stadsfietsen rijden waar er zelfs amper een remvermogen te bespeuren is. Daarnaast is het onderhoud ook veel makkelijker aangezien het minder verduurt door modder, regen of vuil. Dit is dan ook de logische verklaring waarom er in het topwielrennen een overstap naar schijfremmen gebeurt op de weg en er al reeds is in het veld. Exacte cijfers tussen het verschil van schijfremmen of gewone remmen bestaan er niet, maar uit eigen ondervinding gaat mijn voorkeur uit naar schijfremmen als ex-wielrenner.

4.3 Technologie

Het systeem maakt gebruik van de nieuwste technologieën, hierdoor heb je heel wat mogelijkheden. Maar het voornaamste aspect van dit systeem is zijn snelheid en performantie. Met dit systeem is het veel makkelijker om een fiets te huren of verhuren. In deze tijden van nieuwe technologieën vervalt het administratieve werk om een fiets te huren of verhuren naar een geautomatiseerd systeem. Wat het gemak van de gebruiker bevordert alsook de snelheid. Het verdwijnen van al het papierwerk is een drempel dat heel wat gebruikers over de streep zal trekken om dit systeem ook effectief te gaan gebruiken.

Daarnaast biedt deze nieuwe technologie ook de mogelijkheid om gebruik te maken van analyserende software. Hierbij wordt er bijvoorbeeld bijgehouden wat de meest voorkomende fietsroutes zijn en waar er het meest een fiets wordt achtergelaten. Dit kan in het voordeel spelen van de stad, overheidsdiensten of adverteerders. Zo kan bijvoorbeeld bij een nieuw circulatieplan in Gent op basis van deze gegevens rekening gehouden worden met de fietsgevoelige punten in Gent. Samen met het plaatsen van fietsenstallingen op plaatsen waar het zeker nodig is. Dit bevordert het fietsgenot voor de gebruikers alsook voor andere gebruikers van het verkeer. Adverteerders kunnen gebruik maken van deze gegevens om zo hun marketingstrategie aan te passen. Op plaatsen waar veel gebruikers passeren kunnen zij bijvoorbeeld een reclameactie op punt stellen voor de specifieke doelgroep, namelijk de studenten.

5. Verbeterpunten

In dit hoofdstuk sommen we de verbeterpunten van het systeem op. Aangezien dit maar een prototype is, zijn er nog heel wat verbeterpunten op te merken. Dit systeem is opgesteld uit componenten waarvan men dacht dat het de beste oplossing was. Maar eens men het totaalbeeld bekijkt, dan kunnen we al snel vaststellen dat het niet de ideale oplossing is. De verbeteringen op de hardware heeft vooral te maken met het feit dat dit systeem gefinancieerd wordt door mezelf. Hierdoor werd het systeem zo goedkoop mogelijk samengesteld. Wat tevens ook een van de bedoelingen was voor dit systeem. Want mensen zullen eerder geneigd zijn om zo'n goedkoop mogelijke oplossing te kiezen. De verbeterpunten qua software zouden kunnen vermeden worden als er een meer ervaren inkijk op de oplossing zou komen. Als laatstejaars student beschik ik niet over de juiste ervaring om een foutloos en een zo efficiënt mogelijk software oplossing te schrijven.

Met dit hoofdstuk wil ik ook aantonen dat dit systeem nog niet op punt staat. Het gaat hier dan ook om een prototype waarop verder gewerkt kan worden of kan dienen als voorbeeld voor een betere oplossing. De verbeterpunten kunnen we ook beschouwen als nadelen en deze tegenover de voordelen uit hoofdstuk 4 opwegen. Ik noem dit hoofdstuk bewust verbeterpunten en niet nadelen omdat ik er van overtuigd ben dat deze opgelost kunnen worden. Zo kan men deze punten uitwerken en het prototype naar het volgende station brengen tot deze productiewaardig is. Aangezien dit systeem nog niet in de praktijk uitgetest is, zijn nog niet alle verbeterpunten ontdekt. Daardoor is dit hoofdstuk niet compleet maar wel een goede referentie om het protoype op te waarderen als men deze verbeteringen toepast.

5.1 Raspberry Pi 3B

Als basiscomponent voor het systeem heb ik een Raspberry Pi 3B gekozen. Omdat dit een veelzijdige en kleine computer is met een lage kost. Na dit onderzoek kan ik de Raspberry Pi 3B ook makkelijk voor allerhande andere projecten gebruiken. De keuze hiervoor was dus snel gemaakt als men kijkt naar de kost en het toekomstige gebruik. Maar dit is voor dit onderzoek echter niet de ideale oplossing. Eerst en vooral kan het formaat veel compacter en heeft de Raspberry Pi 3B te veel capaciteiten. De verbeterde versie van de Raspberry Pi is eigenlijk een kleine printplaat die ervoor zorgt dat er de vergrendelmodule bestuurt kan worden en dat er een connectie kan gemaakt worden door middel van Bluetooth 4.0.

Allereerst kunnen we een nieuwe oplossing zoeken voor het formaatprobleem. De afmetingen van de Raspberry Pi 3B is zoals aangegeven op de documentatie RaspberryPi (2015) 85mm x 56mm x 15mm. Wat zelf al een klein formaat is als je weet dat de Raspberry Pi kan beschouwd worden als een volwaardige computer. Een versimpeld model wel te verstaan. Als we dan naar andere oplossingen gaan kijken, moeten we wel rekening houden met de functionaliteiten die het zal moeten uitvoeren. Zo zal het verbeterde component een kleine servomotor moeten besturen, alsook een Node.js BLE server kunnen hosten waarop connectie kan gemaakt worden door middel van Bleutooth 4.0.

Het allerkleinste alternatief dat je op de markt vind, is de VoCore2 Ulimate, met een dimensie van 28mm x 30mm x 30mm. Ook de specificaties die men kan vinden op de officiële website van VoCore, voldoen aan de benodigdheden van dit systeem. Tevens is de prijs van de VoCore2 Ultimate gelijkaardig aan de Rapsberry Pi 3B. De VoCore is ontstaan door middel van crowdfunding op Indiegogo door de Chinees Qin Wei. Het doel werd met 1937% gehaald, wat wil zeggen dat van de origneel benodigde $6000 maar liefst $116.194 werd opgehaald ,Indiegogo (2017). Ondanks dat de VoCore2 Ultimate in zijn kinderschoenen staat, is er wel een goede basis. Daarnaast heb je ook nog de flinterdunne CHIP Pro, CHIP, 2017. Het grote nadeel van de CHIP Pro zijn de expansieborden. Per functionaliteit dat je wil toevoegen, moet je een printplaat met die functionaliteit toevoegen. Hierdoor wordt de CHIP Pro alsmaar groter en dus niet ideaal. Maar daarom wil dat niet zeggen dat het geen waardige vervanger is van de huidige Raspberry Pi 3B.

Als we een kijkje nemen op Kickstarter, op zoek naar vernieuwende Internet of Things oplossingen. Dan vinden we al snel de Onion Omega2 ,Onion (2017). Met de grootte van een kers, is dit wel mogelijk de kleinste mogelijke vervanger van de Raspberry Pi 3B te kunnen zijn. De Onion Omega2 is met zijn $ 5 naast de kleinste ook de goedkoopste oplossing voor dit systeem. De Onion Omega2 is een klant-en-klare Linux computer die alle specificaties bevat dat ons systeem nodig heeft. Hierdoor kunnen we makkelijk een Node.js server draaien met connectie via Bleutooth 4.0. Ondanks dat ook dit project heel wat meer geld verzamelde via Kickstarter dan het gewenst had, is dit systeem nog niet ideaal omdat het hoogstwaarschijnlijk toch nog wat kinderziektes bevat.

5.2 Sterk en stijlvol omhulsel

Momenteel is het uitzicht van het systeem ver van optimaal. Een naakte Raspberry Pi 3B, een printplaat en een wirwar van kabels boezemen niet echt vertrouwen uit in een systeem. Niet alleen voor het uitzicht maar ook voor de duurzaamheid van het systeem zou het aan te raden zijn om een sterke en stijlvolle case te maken. Daarnaast is ook het optische aspect van groot belang want een groot deel van de consumenten kiest naast de kostprijs ook op basis van uiterlijk. Als je het beste en sterkste systeem hebt ontwikkeld en je kan je noch onderscheiden op prijs noch op uiterlijk van het systeem, dan is de doorsnee consument niet bereid om het te gebruiken. De doorsnee consument kijkt in een eerste oogopslag niet verder dan de kostprijs of het uiterlijk. Zo wordt het moeilijk om je eerste positieve feedback de wereld in te sturen.

Het is dus belangrijk om een sterke en stijlvolle case te maken. Hiervoor zijn er verschillende opties mogelijk. Eerst en vooral moet er een keuze gemaakt worden tussen een interne bevestiging van het systeem of externe bevestiging op de fiets. De eerste optie is niet optimaal als je verder denkt over hoe de bevestiging in elkaar zal zitten. Zo stuit je op verschillende problemen. Eerst en vooral, waar plaats je het grootste component, met name de Raspberry Pi 3B, zodat het zo vrij van vuil en warmte kan functioneren. Als je een interne bevestiging verkiest, moet je ook gaten voorzien om alle kabels naar buiten te geleiden. En wat als de Raspberry Pi 3B door een defect vervangen moet worden, dan moet je een makkelijk bereikbare plaats kiezen. Alsook is het moeilijk om in een fietskader bevestigingspunten aan te maken. Het enige grote voordeel van een interne bevestiging is de beveiliging tegen diefstal en bescherming tegen vuil en water. Daarnaast moet de dynamo bereikbaar zijn aan het wiel zodat deze stroom kan genereren. En moet het zonnepaneel zo geplaatst worden, dat het zonlicht kan ontvangen alsook stroom opwekken. Bij een interne bevestinging wordt dit alsmaar moeilijker.

Bij de externe bevestiging wordt het grote voordeel van interne bevestiging hier de grootste zorg. Bij externe bevestiging is de Raspberry Pi 3B meer blootgesteld aan de natuurelementen en dus zwakker voor vuil, water en warmte. Dit kan opgelost worden door een case te ontwerpen dat zon- en waterafstotend is. Daarnaast kan de case ook afgekoeld worden tijdens het fietsen door hem zo te bevestigen dat het in de wind gemonteerd staat. Op die manier kan je de frisse wind die je op je gezicht of handen krijgt bij het fietsen op de Raspberry Pi 3B overbrengen. Eens je een sterke case hebt, moet je deze enkel nog op een goede plaats monteren. Deze plaats moet makkelijk bereikbaar zijn indien de Raspberry Pi 3B vervangen moet worden. Ook mag de montage van de Raspberry Pi 3B de fietser niet hinderen tijdens het fietsen. En als laatste, moeten de kabels die naar de vergrendelmodule, de verlichting en de stroomopwekkende componenten goed op het kader kunnen aansluiten zonder te lang te zijn. Bij een langere kabellengte verliest men namelijk stroom en uitvoeringstijd van een bepaalde functie. Als we dit allemaal in acht nemen dan blijven er maar enkele plaatsen meer over. Zo kan je bijvoorbeeld de Rapsberry Pi centraal onder de bovenbuis monteren. Op deze manier is de Raspberry Pi 3B centraal gelegen en bereiken alle kabels de optimale kabellengte en hindert het de fietser niet. Als bevestigingspunt kan men dan een kooi lassen op de fietskader waar de case in past.

5.3 Batterij

Zoals bij heel wat mobiele electronische apparaten is de batterij een heikel punt. Het gebruik van batterijen heeft zo zijn voor- en nadelen. Zo kost het initieel meer om een batterij te ontwerpen in plaats van stroom uit de stekker te gebruiken. Daarnaast heb je ook meer plaats nodig om de batterij te kunnen plaatsen waardoor er eventueel een extra infrastructuur moet voorzien worden. Ten slotte gaat er bij een batterij altijd wel energie verloren door inefficiëntie. Als men stroom uit de stekker gebruikt dan maakt dit stroomverlies niet heel veel uit want je hebt 'oneindige' energietoevoer. Daarnaast vervallen alle nadelen van een batterij bij het gebruik van stroom uit de stekker. Een batterij staat ook niet open voor een slim geïntegreerd energiesysteem om de energie zo veel mogelijk te benutten. Batterijen zijn ook statische objecten waardoor men deze eenmaal maakt en daarna wordt de efficiëntie niet meer verbeterd tenzij men weer een nieuwe batterij maakt. Bij een batterij kan je ook moeilijker gebruik maken van hernieuwbare energie alhoewel dit in dit systeem toch wordt toegepast.

Dit systeem staat ook alleen maar open voor batterijen. Het zou heel onhandig zijn om de fietsen de verzien van een lang kabelsnoer om deze zo van energie te verzien. Je kan jezelf al enkele scenario's inbeelden waardoor dit niet de ideale oplossing is. Daarom maken we momenteel gebruik van de RPi PowerPack v1.2. Deze 3800mAh batterij levert stroom gedurende 9 à 10 uur en heeft een oplaadtijd van ongeveer 3 uur. Op het eerste zicht is dit wel een goede batterij als je weet dat deze een mini computer van stroom kan voorzien. Maar toch is deze batterij te klein voor het systeem. Het systeem wordt namelijk gebruikt door studenten in Gent. Hierbij moet er om de 9 uur minstens 3 uur gefietst worden, als er geen zon is om het zonnepanneel van stroom te voorzien. Deze kans is op zich al relatief klein. En deze wordt alsmaar kleiner als je weet dat studenten in het weekend en 's nachts huiswaarts trekken en er dus makkelijk een tijdspanne van 9 uur over de batterij heen gaat. Heel wat fietsen zouden dus met een lege batterij blijven staan waardoor het systeem niet meer gebruikt kan worden en dus verwaterd in de maatschappij. Op deze manier bieden we zeker geen oplossing op het probleem dat zich nu voordoet. Integendeel we stallen nog meer ongebruikte fietsen op de straathoeken van Gent.

Hier bestaan natuurlijk wel oplossingen voor maar de juiste oplossing kiezen is een ander paar mouwen. Laten we het aantal oplossingen naar 2 vereenvoudigen. De eerste mogelijke oplossing is het gebruik maken van de aan/uit-knop op de RPi PowerPack v1.2. Hierdoor kan de batterij gespaard worden als het systeem lang niet gebruikt moet worden. Ondanks de eenvoudigheid van deze oplossing, is ze niet echt optimaal. Hiervoor zijn er verschillende redenen. Eerst en vooral als de batterij uitstaat, staat het volledige systeem uit en kan het dus niet op de mobiele applicatie gevonden worden. Je moet al een fiets tegenkomen en merken dat het systeem uit staat om hem vervolgens aan te zetten en op die manier terug in werking te brengen. Daarnaast wordt het systeem minder makkelijk door het toevoegen van een extra handeling in het gebruik. Hierdoor wordt het systeem een minder sterke verbetering van de bestaande systemen.

De meest logische oplossing is het monteren van een batterij met meer inhoud. Dit brengt wel enkele moeilijkheden met zich mee. Zo zal de grootte van de batterij logischerwijs toenemen. Waardoor er meer ruimte moet voorzien worden en de oplaadtijd van een volle batterij ook vergroot. In principe kan je zoveel als nodig RPi PowerPack v1.2 aan elkaar koppelen tot je voldoende capaciteit bezit om het systeem van stroom te voozien. Als je het systeem 3 dagen van stroom wil voorzien, wat betekent een capaciteit van 72 uur. Dan moet je minstens 8 RPi PowerPack v1.2's in serie koppelen. Wat ook wil zeggen dat het systeem 8 keer groter wordt. Het ontwerp van een batterij op maat is dus aangeraden. Maar de RPi PowerPack v1.2 kan wel de ideale batterij zijn, als een van de voorgaande verbeterpunten wordt doorgevoerd. Met name het veranderen van de Raspberry Pi 3B door een energiezuiniger alternatief. De uptime van het alternatief moet wel getest worden om vervolgens te kunnen uitmaken of de RPi PowerPack v1.2 voldoet of niet.

5.4 Duurzaamheid en beveiliging

Tijdens het maken van dit systeem was de duurzaamheid van de vergrendelmodule mijn grootste zorg. Het ontwikkelen van goede automatische vergrendeling was geen sinecure. Het uitlijnen van de zelfgemaakte tandwielen was ver van makkelijk en kan dus beter. De tandwielen moeten nauwkeuriger gemaakt worden en dit kan door middel van een 3D printer die tot op de mm nauwkeurig de tandenwielen maakt. Hierdoor lijnen ze beter uit en wordt de vergrendelmodule algemeen sterker. Daarnaast moet er ook een case komen op de vergrendelmodule zodat deze bestand is tegen water en vuil. Op de schijfremmen kan Brake Protec gemonteerd worden, BrakeProtec (2017). Brake Protec zorgt ervoor dat de schrijfremmen schoon blijven waardoor de remblokken langer meegaan en ook de vergrendelmodule vrij blijft van vuil. Alsook kan de servomotor wel een upgrade gebruiken. Als initieel voorbeeld voldoet de motor aan de verwachtingen. Maar als men het prototype uitwerkt, dan zou een nauwkeurigere en stabielere servomotor wel op zijn plaats zijn. Door deze aanpassingen kan de vergrendelmodule in eerste instantie versterkt en dus duurzamer worden voor gebruik.

Het systeem moet niet alleen ideaal zijn voor de consument maar ook voor de verdeler. Het systeem is momenteel makkelijk te demonteren. Hierdoor kan je bijvoorbeeld makkelijk een Raspberry Pi 3B bemachtigen. Niet echt ideaal dus om als producent het systeem de wereld in te sturen met een heel grote kans dat het systeem niet gebruikt wordt waarvoor het bedoeld is maar voor de onderdelen dat het bevat. Dit probleem kan deels opgelost worden door een voorgaand verbeterpunt mee te verwerken in dit probleem. Eerder hebben we al aangehaald hoe de case van het systeem moet voldoen aan bepaalde voorwaarden zodat het stijlvoller is en bestand tegen verschillende weersomstandigheden. Aan de voorwaarden kunnen we ook nog beveiliging van de componenten toevoegen. Je kan de case meer beveiligen door er een slot met sleutel eraan toe te voegen of door gebruik te maken van klinknagels om hem aan het frame te bevestigen. Op deze 2 manieren kunnen de componenten nog vervangen worden bij defect en is de beveiliging van de componenten verhoogd.

Als laatste verbeterpunt wil ik de beveiling tegen aanvallen op de software aanhalen. Ondanks dat de mogelijkheid tot connectie wegvalt als er al een connectie gemaakt is, zal er iemand dit wel kunnen omzeilen. Momenteel worden er simpele berichten verstuurt met de boodschap open of close van de mobiele applicatie naar de Bluetooth Low Energy server op de Raspberry Pi 3B. Nu je dit weet, is het makkelijk om zonder de speciaal ontworpen mobiele applicatie, de Bluetooth Low Energy server aan te spreken, door bijvoorbeeld nRF Connect ,Nordic-semiconductor (2017). Met deze mobiele applicatie kan je zelf in te vullen berichten via bluetooth verzenden. Als je dan weet dat de inhoud van zo'n bericht voor dit systeem simpelweg öpenöf "closeïs, is het systeem makkelijk te omzeilen. Dit valt op te lossen door de berichten te encrypten en op deze manier te verzegelen.

6. Uitwerking

In dit hoofdstuk overlopen we kort de uitwerking van het systeem. Hierbij bekijken we de code en technologie die gebruikt werd om de verschillende componenten op te zetten. Eerst bekijken we de componenten die gebruikt worden op de Raspberry Pi 3B. We kijken hoe de Node.js Bleutooth Low Energy server is opgesteld en hoe deze zich open stelt voor communicatie met de Android mobiele applicatie. Vervolgens bespreken we de vergrendelmodule waarbij het script om de servomotor aan te sturen van groot belang is. Naast de Raspberry Pi 3B hebben we nog heel wat code te bespreken voor de Android mobiele applicatie. Hierbij geef ik graag wat extra uitleg over de het Login-mechanisme en de real-time database die gemaakt werden met FireBase. Tot slot bekijken we het belangrijkste deel van de Android mobiele applicatie waarbij we de code van de Bleutooth Low Energy onder de loep nemen.

6.1 Raspberry Pi 3B

6.1.1 Bluetooth Low Energy server

De Bleutooth Low Energy server bestaat uit een Node.js server met 2 javascript bestanden. De 'main.js' zorgt voor de opstart van bleno op de Node.js server. De bleno nodemodule wordt opgeroepen, die op zijn beurt een PrimaryService oproept. Daarna start bleno zijn advertising waarin een naam een UUID wordt meegegeven. Binnen de advertising methode worden de services gezet, hier wordt een PrimaryService gezet met de YanCharacteristic. De inhoud van deze YanCharacteristic vinden we in de 'yanCharacteristic.js'.

In dit bestand wordt een BlenoCharacteristic gemaakt met 'write' properties. Dit wil zeggen dat er enkel naar deze module geschreven kan worden en er geen antwoord terug verwacht wordt. In de 'onWriteRequest' methode wordt er geluisterd naar de waarde die naar de module geschreven wordt. Als deze waarde 'open' of 'close' bevat, dan wordt er een bijhorend Python script aangeroepen voor de vergrendelmodule.

Listing 6.1: main.js

```
var bleno = require('bleno');
var BlenoPrimaryService = bleno.PrimaryService;
var YanCharacteristic = require('./yanCharacteristic');

console.log('start bleno for yan');

bleno.on('stateChange', function(state) {
        console.log('on -> stateChange: ' + state);
        if (state === 'poweredOn') {
            bleno.startAdvertising('yan', ['1a970c92
               -4559-11e7-a919-92ebcb67fe33']);
        } else {
            bleno.stopAdvertising();
        }
    });

    bleno.on('advertisingStart', function(error) {
        console.log('on -> advertisingStart: ' + (error ? '
           error' : 'succes'));
        if (!error) {
            bleno.setServices([
                new BlenoPrimaryService({
                    uuid: '795090c7-420d-4048-a24e-18
                       e60180e23c',
                    characteristics: [
                        new YanCharacteristic()
                    ]
                })
            ]);
        }
    });
```

Listing 6.2: yanCharacteristic.js

```
var util = require('util');
var bleno = require('bleno');
var PythonShell = require('python-shell');

var BlenoCharacteristic = bleno.Characteristic;

var YanCharacteristic = function() {
  YanCharacteristic.super_.call(this, {
    uuid: '0b89d2d4-0ea6-4141-86bb-0c5fb91ab14a',
    properties: ['write'],
    value: null
  });
```

```
  this._value = new Buffer(0);
  this._updateValueCallback = null;
};

util.inherits(YanCharacteristic, BlenoCharacteristic);

YanCharacteristic.prototype.onWriteRequest = function(data,
    offset, withoutResponse, callback) {
  this._value = data;
  console.log('YanCharacteristic - onWriteRequest: value = '
     + this._value.toString());
  if (this._value.toString() == "open") {
    console.log('Lock will open');
    PythonShell.run('servoOpen.py', function(err) {
      if (err) throw err;
      console.log('finished');
    });
  }

  if (this._value.toString() == "close") {
    console.log('Lock will close');
    PythonShell.run('servoClose.py', function(err) {
      if (err) throw err;
      console.log('finished');
    });
  }

  if (this._updateValueCallback) {
    console.log('YanCharacteristic - onWriteRequest : '
       notifying');
    this._updateValueCallback(this._value);
  }
  callback(this.RESULT_SUCCESS);
};

module.exports = YanCharacteristic;
```

6.1.2 Vergrendelmodule

De vergrendelmodule bestaat uit 2 Python script die communiceren met de GPIO van de Raspberry Pi 3B. Deze scripten zijn genaamd als 'servoClose.py' en 'servoOpen.py', respectievelijke zorgen zij voor het sluiten en openen van de vergrendelmodule. Eerst wordt de Raspberry Pi 3B GPIO ingeladen zodat er interactie kan gemaakt worden met de invoer- en uitvoerpinnen. Eerst wordt de GPIO-mode op BOARD gezet zodat er gewerkt kan worden met de nummering van de pinnen. Daarna wordt de 7de pin ingesteld als uitvoerpin, hierdoor kunnen er signalen op pin 7 uitgevoerd worden. Vervolgens wordt de pulsbreedte van de uitvoerpin 7 ingesteld op 50 Hertz. Om de servomotor dan te besturen wordt de startpositie ingesteld, gevolgd door een 'ChangeDutyCycle' waarbij de servomotor naar zijn eindpositie wordt gebracht.

Listing 6.3: servoOpen.py

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
p = GPIO.PWM(7,50)
p.start(2.5)
p.ChangeDutyCycle(12.5)
```

Listing 6.4: servoClose.py

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
p = GPIO.PWM(7,50)
p.start(12.5)
p.ChangeDutyCycle(2.5)
```

6.2 Android mobiele applicatie

6.2.1 Login

Om de authenticatie binnen de Android mobiele applicatie op te stellen, moeten we eerst de juiste software development kits toevoegen. Bij ons gaat dat om de 'firebase-auth' dependency. Daarna kunnen we onze Login activiteit opstellen. Deze zal zorgen voor een simpele e-mail met wachtwoord authenticatie. Eerst en vooral wordt er een FirebaseAuth object aangemaakt, hiermee kunnen we communiceren met Firebase Authentication en kunnen we de toestand van de gebruiker bekijken. Firebase biedt zowel een 'create' als 'signIn' functie aan om gebruikers te laten registreren en inloggen op de Android mobiele applicatie. Verder wordt het login scherm opgesteld door enkele textview's , editText's en buttons.

Listing 6.5: LoginActivity

```
public class LoginActivity extends Activity implements View
    .OnClickListener {

    private static final String TAG = "LOGIN";
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener mAuthListener;

    private EditText emailField, passwordField;
    private TextView changeSignUpModeTextView;
    private boolean signUpActive;
    private Button signUpButton;

    private FirebaseDatabase database = FirebaseDatabase.
       getInstance();
    private DatabaseReference myRef = database.getReference
       ("database");
    private ArrayList<String> emails = new ArrayList<>();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        emailField = (EditText) findViewById(R.id.
           emailEditText);
        passwordField = (EditText) findViewById(R.id.
           passwordEditText);
        changeSignUpModeTextView = (TextView) findViewById(
           R.id.logInTextView);
        signUpActive = true;
        signUpButton = (Button) findViewById(R.id.
           signInButton);
        signUpButton.setOnClickListener(new View.
           OnClickListener() {
            @Override
            public void onClick(View v) {
                signUpOrLogin(v);
            }
        });

        changeSignUpModeTextView.setOnClickListener(this);

        mAuth = FirebaseAuth.getInstance();
```

```
        mAuthListener = new FirebaseAuth.AuthStateListener
            () {
            @Override
            public void onAuthStateChanged(@NonNull
                FirebaseAuth firebaseAuth) {
                FirebaseUser user = firebaseAuth.
                    getCurrentUser();
                if (user != null) {
                    Log.d(TAG, "onAuthStateChanged:␣signed␣
                        in");
                } else {
                    Log.d(TAG, "onAuthStateChanged:␣signed␣
                        out");
                }
            }
        };
    }

    @Override
    protected void onStart() {
        super.onStart();
        mAuth.addAuthStateListener(mAuthListener);
    }

    @Override
    protected void onStop() {
        super.onStop();
        mAuth.addAuthStateListener(mAuthListener);
    }

    public void createAccount(final String email, String
        password) {
        myRef.child("users").addValueEventListener(new
            ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot
                dataSnapshot) {
                for (DataSnapshot dataSnapshot1 :
                    dataSnapshot.getChildren()) {
                    if (email.equals(dataSnapshot1.getValue
                        ().toString())) {
                        Toast.makeText(
                            getApplicationContext(), "Email␣
                            already␣exists", Toast.
                            LENGTH_SHORT).show();
                        signUpActive = false;
```

```
                    changeSignUpModeTextView.setText("
                        Sign Up");
                    signUpButton.setText("Log In");
                    return;
                }
            }
            Log.i(TAG, "onDataChange: " + emails.
                toString());
        }

        @Override
        public void onCancelled(DatabaseError
            databaseError) {
            Toast.makeText(getApplicationContext(), "
                Failed to read data", Toast.LENGTH_SHORT
                ).show();
        }
    });


    mAuth.createUserWithEmailAndPassword(email,
        password)
            .addOnCompleteListener(this, new
                OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<
                    AuthResult> task) {
                    Log.d(TAG, "onComplete: create " +
                        task.isSuccessful());

                    if (!task.isSuccessful()) {
                        Toast.makeText(LoginActivity.
                            this, "creation failed",
                            Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(LoginActivity.
                            this, "creation of user is
                            successful", Toast.
                            LENGTH_SHORT).show();
                        FirebaseUser userFirebase =
                            mAuth.getCurrentUser();
                        myRef.child("users").push().
                            setValue(email);
                        Log.i(TAG, "CurrentUser
                            creation: " + userFirebase.
                            getEmail());
```

```
                    }
                }
            });
    }

    public void signIn(String email, String password) {
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
                    OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<
                        AuthResult> task) {
                        Log.d(TAG, "onComplete:␣login␣" +
                            task.isSuccessful());
                        if (!task.isSuccessful()) {
                            Toast.makeText(LoginActivity.
                                this, "login␣failed", Toast.
                                LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(LoginActivity.
                                this, "user␣is␣logged␣in",
                                Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }

    public void signUpOrLogin(View view) {
        Log.i(TAG, "signUpOrLogin:␣entering");
        String email = String.valueOf(emailField.getText())
            ;
        String pass = String.valueOf(passwordField.getText
            ());
        if (email.equals("") || pass.equals("")) {
            Toast.makeText(LoginActivity.this, "Please␣fill
                ␣in␣the␣fields", Toast.LENGTH_SHORT).show();
        } else {
            Pattern p = Pattern.compile(".+@.+\\.[a-z]+");
            Matcher m = p.matcher(email);
            boolean matchFound = m.matches();
            if (!matchFound) {
                Toast.makeText(LoginActivity.this, "Please␣
                    enter␣a␣valid␣email", Toast.LENGTH_SHORT
                    ).show();
                if (pass.length() < 6) {
```

```
                    Toast.makeText(LoginActivity.this, "
                        Password must have more than 6
                        digits", Toast.LENGTH_SHORT).show();
                }
            } else {
                if (signUpActive) {
                    createAccount(email, pass);

                } else {
                    signIn(email, pass);
                }
            }
        }
    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.logInTextView) {
            Log.i(TAG, "onClick: loginText");
            if (signUpActive) {
                signUpActive = false;
                changeSignUpModeTextView.setText("Sign Up")
                    ;
                signUpButton.setText("Log In");

            } else {
                signUpActive = true;
                changeSignUpModeTextView.setText("Log In");
                signUpButton.setText("Sign Up");
            }
        }
    }
}
```

6.2.2 Database

Gelijklopend met de Firebase Authentication kunnen we ook Firebase Database makkelijk toevoegen aan onze Android mobiele applicatie. Net zoals bij authenticatie moet voor de database de juiste dependency toegevoegd worden aan het project, hier is dat dat 'firebase-database' dependency. Daarna moet je enkel nog de schrijf- en leesrechten van de database configureren en je bent klaar om de database te gebruiken. Door een 'instance' van het FirebaseDatabase object aan te roepen, kan je refereren naar de locatie waar je je data wil opslaan of wijzigen. Door de simpele 'getValue' en 'setValue' kan je de waarden van je objecten verkrijgen of wijzigen. In dit systeem slaan we alle gebruikers op in de database alsook houden we de logs bij wanneer de open en close operaties worden aangeroepen.

Listing 6.6: Gebruikers toevoegen aan database

```
private FirebaseDatabase database = FirebaseDatabase.
    getInstance();
private DatabaseReference myRef = database.getReference("
    database");
/../
myRef.child("users").push().setValue(email);
```

Listing 6.7: Gebruikers ophalen uit database

```
private FirebaseDatabase database = FirebaseDatabase.
    getInstance();
private DatabaseReference myRef = database.getReference("
    database");
/../
 myRef.child("users").addValueEventListener(new
     ValueEventListener() {
             @Override
             public void onDataChange(DataSnapshot
                 dataSnapshot) {
                 for (DataSnapshot dataSnapshot1 :
                     dataSnapshot.getChildren()) {
                     if (email.equals(dataSnapshot1.getValue
                         ().toString())) {
                         Toast.makeText(
                             getApplicationContext(), "Email␣
                             already␣exists", Toast.
                             LENGTH_SHORT).show();
                         signUpActive = false;
                         changeSignUpModeTextView.setText("
                             Sign␣Up");
                         signUpButton.setText("Log␣In");
                         return;
                     }
                 }
                 Log.i(TAG, "onDataChange:␣" + emails.
                     toString());
             }

             @Override
             public void onCancelled(DatabaseError
                 databaseError) {
                 Toast.makeText(getApplicationContext(), "
                     Failed␣to␣read␣data", Toast.LENGTH_SHORT
                     ).show();
             }
         });
```

Listing 6.8: Logs toevoegen aan database

```
private FirebaseDatabase database = FirebaseDatabase.
    getInstance();
private DatabaseReference reference = database.getReference
    ("database");
/../
String log = String.format("opening␣"+ deviceAddress .
    toString() +"␣by␣user␣=␣test-user␣on␣" +
    currentDateandTime);
reference.child("logs").push().setValue(log);
/../
String log = String.format("closing␣"+ deviceAddress.
    toString()+"␣by␣user␣=␣test-user␣on␣" +
    currentDateandTime);
        reference.child("logs").push().setValue(log);
```

6.2.3 Bluetooth Low Energy

Om Bluetooth Low Energy mogelijk te maken binnen Android, heb ik de Android BLE Scanner Compat library van Nordic Semiconductor toegevoegd. Deze library lost het probleem op waarbij zoeken naar Bluetooth Smart apparaten op Android niet lukt. Deze library overschrijft heel wat standaard libraries waardoor zoeken naar Bluetooth Low Energy apparaten mogelijk is. Eerst en vooral hebben we de DeviceScanActivity, hierin zoeken we naar Bleutooth Low Energy apparaten waarvan het UUID gelijk is aan het UUID van de Node.js Bluetooth Low Energy server op de Raspberry Pi 3B. De Activity zoekt 10 seconden naar mogelijke apparaten en plaatst de gevonden appareten in een lijst die op het scherm getoond wordt. Als er op een van de apparaten uit de lijst gedrukt wordt, dan opent het de DeviceControlAcitivity.

Deze DeviceControlAcitivity opent een scherm met daarop basisinformatie van het apparaat waarmee connectie zal gemaakt worden. Deze basisinformatie bestaat uit het adres en de naam van het apparaat. Onder de basisinformatie vind je 2 knoppen. Deze knoppen stellen de open en close operaties voor die al eerder vermeld werden. Deze knoppen starten hun eigen operatie die op zijn beurt de juiste informatie doorspeelt aan de server. Deze overdracht van informatie gebeurt aan de hand van een GattClient. Deze client maakt connectie met het Bluetooth Low Energy apparaat en beschikt over de juiste services en characteristics. Aan de hand daarvan kunnen we kleine hoeveelheden data doorsturen, in dit geval de tekst open of close. Deze data wordt naar de server geschreven. De server ontvangt de data en roept aan de hand van deze data de juiste operatie aan.

Listing 6.9: DeviceScanActivity

```
public class DeviceScanActivity extends AppCompatActivity {

    private static final String TAG = DeviceScanActivity.
       class.getSimpleName();
    private boolean scanning;
    private ListView listView;
    private ArrayList<String> deviceList = new ArrayList
       <>();
    private ArrayList<BluetoothDevice> devices = new
       ArrayList<>();
    private ArrayAdapter<String> adapter;

    private static final int REQUEST_ENABLE_BT = 1;
    private static final long SCAN_PERIOD = 10_000;
    private static final int REQUEST_PERMISSION_LOCATION =
       1;

    private static final UUID SERVICE_UUID = UUID.
       fromString("1a970c92-4559-11e7-a919-92ebcb67fe33");

    private final BluetoothLeScannerCompat scanner =
       BluetoothLeScannerCompat.getScanner();
    private final Handler stopScanHandler = new Handler();
    private final Runnable stopScanRunnable = new Runnable
       () {
        @Override
        public void run() {
            if (deviceList.isEmpty()) {
                Toast.makeText(getApplicationContext(), "No
                   devices found", Toast.LENGTH_SHORT).
                   show();
            }
            stopLeScan();
        }
    };

    private final ScanCallback scanCallback = new
       ScanCallback() {
        @Override
        public void onScanResult(int callbackType,
           ScanResult result) {
            Log.i(TAG, "onScanResult: " + result.getDevice
               ().getAddress());

        }
```

```
        @Override
        public void onBatchScanResults(List<ScanResult>
            results) {
            Log.i(TAG, "onBatchScanResults:␣" + results.
                toString());

            for (ScanResult result : results) {
                if (!deviceList.contains(result.getDevice()
                    .getAddress())) {
                    deviceList.add(result.getDevice().
                        getAddress());
                    devices.add(result.getDevice());
                    adapter.notifyDataSetChanged();
                }
            }
            Log.i(TAG, "devicelist" + deviceList.toString()
                );
        }


        @Override
        public void onScanFailed(int errorCode) {
            Log.w(TAG, "onScanFailed:␣" + errorCode);
            stopLeScan();
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_scan);
        listView = (ListView) findViewById(R.id.list);
        adapter = new ArrayAdapter<>(this, android.R.layout
            .simple_list_item_1, deviceList);
        adapter.notifyDataSetChanged();
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(new AdapterView.
            OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent,
                View view, int position, long id) {
                final BluetoothDevice device = devices.get(
                    position);
                if (device == null) return;
                startInteractActivity(device);
```

```
            }
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        if (!scanning) {
            menu.findItem(R.id.menu_stop).setVisible(false)
                ;
            menu.findItem(R.id.menu_scan).setVisible(true);
            menu.findItem(R.id.menu_refresh).setActionView(
                null);
        } else {
            menu.findItem(R.id.menu_stop).setVisible(true);
            menu.findItem(R.id.menu_scan).setVisible(false)
                ;
            menu.findItem(R.id.menu_refresh).setActionView(
                R.layout.actionbar_indeterminate_progress);
        }
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_scan:
                startLeScan();
                invalidateOptionsMenu();
                break;
            case R.id.menu_stop:
                stopLeScan();
                break;
        }
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int
        resultCode, Intent data) {
        if (requestCode == REQUEST_ENABLE_BT && resultCode
            == Activity.RESULT_CANCELED) {
            Toast.makeText(this, "You must turn Bluetooth
                on, to use this app", Toast.LENGTH_LONG).
                show();
```

```
            finish();
            return;
        }
        super.onActivityResult(requestCode, resultCode,
            data);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
        String[] permissions, int[] grantResults) {
        if (requestCode == REQUEST_PERMISSION_LOCATION &&
            grantResults.length > 0 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
            Log.i(TAG, "Permission␣accepted");
        } else {
            Toast.makeText(this, "You␣must␣grant␣the␣
                location␣permission.", Toast.LENGTH_SHORT).
                show();
            finish();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        prepareForScan();
    }


    @Override
    protected void onPause() {
        super.onPause();
        stopLeScan();
    }

    private void prepareForScan() {
        if (isBleSupported()) {
            BluetoothManager btManager = (BluetoothManager)
                getSystemService(Context.BLUETOOTH_SERVICE)
                ;
            BluetoothAdapter btAdapter = btManager.
                getAdapter();
            if (btAdapter.isEnabled()) {
                if (ContextCompat.checkSelfPermission(this,
                    android.Manifest.permission.
                    ACCESS_FINE_LOCATION) == PackageManager.
```

```
                PERMISSION_GRANTED) {
                    startLeScan();
                } else {
                    ActivityCompat.requestPermissions(this,
                        new String[]{android.Manifest.
                        permission.ACCESS_FINE_LOCATION},
                        REQUEST_PERMISSION_LOCATION);
                }
            } else {
                Intent enableBluetooth = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBluetooth,
                    REQUEST_ENABLE_BT);
            }
        } else {
            Toast.makeText(this, "BLE is not supported",
                Toast.LENGTH_LONG).show();
            finish();
        }
    }

    private boolean isBleSupported() {
        return getPackageManager().hasSystemFeature(
            PackageManager.FEATURE_BLUETOOTH_LE);
    }

    private void startLeScan() {
        scanning = true;

        ScanSettings settings = new ScanSettings.Builder()
                .setScanMode(ScanSettings.
                    SCAN_MODE_LOW_LATENCY)
                .setReportDelay(1000)
                .build();
        List<ScanFilter> filters = new ArrayList<>();
        filters.add(new ScanFilter.Builder().setServiceUuid
            (new ParcelUuid(SERVICE_UUID)).build());
        scanner.startScan(filters, settings, scanCallback);
        stopScanHandler.postDelayed(stopScanRunnable,
            SCAN_PERIOD);
        invalidateOptionsMenu();
    }

    private void stopLeScan() {
        if (scanning) {
            scanning = false;
```

```
                scanner.stopScan(scanCallback);
                stopScanHandler.removeCallbacks(
                    stopScanRunnable);
                invalidateOptionsMenu();
            }
        }

        private void startInteractActivity(BluetoothDevice
            device) {
            Intent intent = new Intent(this,
                DeviceControlActivity.class);
            intent.putExtra(DeviceControlActivity.
                EXTRAS_DEVICE_NAME, device.getName());
            intent.putExtra(DeviceControlActivity.
                EXTRAS_DEVICE_ADDRESS, device.getAddress());
            startActivity(intent);
            finish();
        }
}
```

Listing 6.10: DeviceControlActivity

```
public class DeviceControlActivity extends
    AppCompatActivity {

    private TextView deviceName,deviceAddress;
    public static final String EXTRAS_DEVICE_NAME = "
        DEVICE_NAME";
    public static final String EXTRAS_DEVICE_ADDRESS = "
        DEVICE_ADDRESS";
    private final GattClient gattClient = new GattClient();
    private Button buttonClose, buttonOpen;
    private FirebaseDatabase database = FirebaseDatabase.
        getInstance();
    private DatabaseReference reference = database.
        getReference("database");

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_control);
        Intent intent = getIntent();
        buttonOpen = (Button) findViewById(R.id.buttonOpen)
            ;
        buttonOpen.setText("open");
        buttonOpen.setEnabled(false);
        buttonOpen.setOnClickListener(new View.
```

```
    OnClickListener() {
     @Override
     public void onClick(View v) {
         openButton();
     }
});

buttonClose = (Button) findViewById(R.id.
    buttonClose);
buttonClose.setEnabled(false);
buttonClose.setOnClickListener(new View.
    OnClickListener() {
     @Override
     public void onClick(View v) {
         closeButton();
     }
});
gattClient.onCreate(this, intent.getStringExtra(
    EXTRAS_DEVICE_ADDRESS), new GattClient.
    OnReadListener() {

     @Override
     public void onConnected(final boolean success)
        {
         runOnUiThread(new Runnable() {
             @Override
             public void run() {
                     buttonOpen.setEnabled(success);
                     if(!success){
                     Toast.makeText(
                         DeviceControlActivity.this,"
                         Connection error",Toast.
                         LENGTH_SHORT).show();
                 }
             }
         });
     }
});

deviceName = (TextView) findViewById(R.id.
    controlDeviceName);
deviceAddress = (TextView) findViewById(R.id.
    controlDeviceAddress);

deviceName.setText(intent.getStringExtra(
    EXTRAS_DEVICE_NAME));
```

```
        deviceAddress.setText(intent.getStringExtra(
            EXTRAS_DEVICE_ADDRESS));
    }

    private void openButton(){
        write("open");
        SimpleDateFormat sdf = new SimpleDateFormat("
            yyyyMMdd_HHmmss");
        String currentDateandTime = sdf.format(new Date());
        String log = String.format("opening␣"+
            deviceAddress .toString() +"␣by␣user␣=␣test-user
            ␣on␣" +currentDateandTime);
        reference.child("logs").push().setValue(log);
        buttonOpen.setEnabled(false);
        buttonClose.setEnabled(true);
    }

    private void closeButton() {
        write("close");
        SimpleDateFormat sdf = new SimpleDateFormat("
            yyyyMMdd_HHmmss");
        String currentDateandTime = sdf.format(new Date());
        String log = String.format("closing␣"+
            deviceAddress.toString()+"␣by␣user␣=␣test-user␣
            on␣" +currentDateandTime);
        reference.child("logs").push().setValue(log);
        gattClient.onDestroy();
        Intent intent = new Intent(getApplicationContext(),
            DeviceScanActivity.class);
        startActivity(intent);
    }

    private void write(String valueString){
        gattClient.writeInteractor(valueString);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        gattClient.onDestroy();
    }
}
```

Listing 6.11: GattClient

```
public class GattClient {
```

```
private static final String TAG = GattClient.class.
    getSimpleName();

public interface OnReadListener {
    void onConnected(boolean success);
}

private Context context;
private OnReadListener listener;
private String deviceAddress;

private static final UUID SERVICE_UUID = UUID.
    fromString("795090c7-420d-4048-a24e-18e60180e23c");
private static UUID CHARACTERISTIC_INTERACTOR_UUID =
    UUID.fromString("0b89d2d4-0ea6-4141-86bb-0
    c5fb91ab14a");

private BluetoothManager manager;
private BluetoothAdapter adapter;
private BluetoothGatt bluetoothGatt;
private BluetoothGattCallback gattCallback = new
    BluetoothGattCallback() {
    @Override
    public void onConnectionStateChange(BluetoothGatt
        gatt, int status, int newState) {
        if (newState == BluetoothProfile.
            STATE_CONNECTED) {
            gatt.discoverServices();
        } else if (newState == BluetoothProfile.
            STATE_DISCONNECTED) {
            listener.onConnected(false);
        }
    }

    @Override
    public void onServicesDiscovered(BluetoothGatt gatt
        , int status) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
            boolean connected = false;

            BluetoothGattService service = gatt.
                getService(SERVICE_UUID);
            if (service != null) {
                if ((service.getCharacteristic(
                    CHARACTERISTIC_INTERACTOR_UUID) !=
                    null)) {
```

```
                    connected = true;
                }
            }
            listener.onConnected(connected);
        } else {
            Log.w(TAG, "onServicesDiscovered␣received:␣
                " + status);
        }
    }
};
private final BroadcastReceiver mBluetoothReceiver =
    new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent
        intent) {
        int state = intent.getIntExtra(BluetoothAdapter
            .EXTRA_STATE, BluetoothAdapter.STATE_OFF);

        switch (state) {
            case BluetoothAdapter.STATE_ON:
                startClient();
                break;
            case BluetoothAdapter.STATE_OFF:
                stopClient();
                break;
            default:
                break;
        }
    }
};

public void onCreate(Context context, String
    deviceAddress, OnReadListener listener) throws
    RuntimeException {
    this.context = context;
    this.listener = listener;
    this.deviceAddress = deviceAddress;

    this.manager = (BluetoothManager) context.
        getSystemService(BLUETOOTH_SERVICE);
    this.adapter = manager.getAdapter();
    if (!checkBluetoothSupport(adapter)) {
        throw new RuntimeException("GATT␣client␣
            requires␣Bluetooth␣support");
    }
```

```
        IntentFilter filter = new IntentFilter(
            BluetoothAdapter.ACTION_STATE_CHANGED);
        this.context.registerReceiver(mBluetoothReceiver,
            filter);
        if (!adapter.isEnabled()) {
            Log.w(TAG, "Bluetooth is currently disabled... "
                enabling");
            adapter.enable();
        } else {
            Log.i(TAG, "Bluetooth enabled... starting "
                client");
            startClient();
        }
    }

    public void onDestroy() {
        listener = null;

        BluetoothAdapter bluetoothAdapter = manager.
            getAdapter();
        if (bluetoothAdapter.isEnabled()) {
            stopClient();
        }

        context.unregisterReceiver(mBluetoothReceiver);
    }

    public void writeInteractor(String value) {
        BluetoothGattService service = bluetoothGatt.
            getService(SERVICE_UUID);
        BluetoothGattCharacteristic interactor = service
                .getCharacteristic(
                    CHARACTERISTIC_INTERACTOR_UUID);
        interactor.setValue(value);
        bluetoothGatt.writeCharacteristic(interactor);
    }

    private boolean checkBluetoothSupport(BluetoothAdapter
        bluetoothAdapter) {
        if (bluetoothAdapter == null) {
            Log.w(TAG, "Bluetooth is not supported");
            return false;
        }

        if (!context.getPackageManager().hasSystemFeature(
            PackageManager.FEATURE_BLUETOOTH_LE)) {
```

```
            Log.w(TAG, "Bluetooth LE is not supported");
            return false;
        }

        return true;
    }

    private void startClient() {
        BluetoothDevice bluetoothDevice = adapter.
           getRemoteDevice(deviceAddress);
        bluetoothGatt = bluetoothDevice.connectGatt(context
           , false, gattCallback);
        if (bluetoothGatt == null) {
            Log.w(TAG, "Unable to create GATT client");
            return;
        }
    }

    private void stopClient() {
        if (bluetoothGatt != null) {
            bluetoothGatt.disconnect();
            bluetoothGatt.close();
        }

        if (adapter != null) {
            adapter = null;
        }
    }

}
```

7. Conclusie

Nu we aan het einde gekomen zijn van deze bachelorproef, wil ik voor mezelf nog eens terugkeren naar het doel van dit onderzoek. Kunnen we door middel van Internet of Things de transportatiemiddelen van studenten verbeteren? Op deze vraag kan ik volmondig ja antwoorden maar deze ja heeft ook zo zijn complicaties. Toch zit er zeker een toekomst in de verbetering van transportatiemiddelen voor studenten en het algemeen vervoer in de stad. Dit onderzoek is een eerste stap naar deze verbetering. Het gaat niet alleen om de makkelijke en efficiënte manier van verplaatsing voor de gebruiker maar ook qua milieuvriendelijkheid voor de stad. Zo zullen er minder ongebruikte fietsen in de stad te vinden zijn en zullen ook de uitlaatgassen dalen door het stijgende gebruik van fietsen in plaats van gemotoriseerde vervoersmiddelen.

Het resultaat van dit onderzoek is meer dan wat ik verwacht had te bereiken. Maar daarom zeker nog niet compleet. Desondanks ben ik tevreden met het prototype dat ik gecreëerd heb. De basisfunctionaliteiten zijn aanwezig en alles kan mooi met elkaar samenwerken. Alsook zijn de verbeterpunten op tafel gelegd en kennen we de hekele punten van het systeem. Het prototype bestaat misschien niet uit de beste kwaliteit maar kan wel dienen als ideale springplank voor verdere projecten. Het toepassen van de verbeterpunten zou van dit systeem een waardig prototype maken om tegen de hedendaagse transportmiddelen te concurreren.

Bibliografie

Android. (2017). Bluetooth Low Energy. Verkregen van https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Bleutooth. (2017). Bluetooth specifications. Verkregen van https://www.bluetooth.com/specifications/adopted-specifications

Blue-bike. (2017). Blue bike tarieven. Verkregen van https://www.blue-bike.be/nl/tarieven

BrakeProtec. (2017). Hoe werkt brake protec. Verkregen van https://brakeprotec.com/start/hoe%20werkt%20de%20brake%20protec.html

CHIP. (2017). CHIP Pro. Verkregen van https://getchip.com/pages/chippro

DeLijn. (2017). Buzzy pas. Verkregen van https://www.delijn.be/nl/vervoerbewijzen/abonnementen/buzzy-pazz.html

Dickey, J. (2017). A Quick Introduction to How Node.js Works. Verkregen van https://dzone.com/articles/quick-introduction-how-nodejs

Firebase. (2017). Firebase. Verkregen van https://firebase.google.com/

Gradle. (2017). Gradle build system. Verkregen van https://gradle.org/

Huang, J. (2016). Road disc brakes: everything you need to know. Verkregen van http://www.bikeradar.com/road/gear/article/disc-brakes-everything-you-need-to-know-bikeradar-45859/

Indiegogo. (2017). VoCore: A coin-sized Linux computer with wifi. Verkregen van https://www.indiegogo.com/projects/vocore-a-coin-sized-linux-computer-with-wifi#/

Java. (2017). What is Java? Verkregen van https://www.java.com/nl/download/faq/whatis_java.xml

JetBrains. (2017). JetBrains IDE's. Verkregen van https://www.jetbrains.com/

Micropik. (2016). SG90 specification guide. Verkregen van http://www.micropik.com/PDF/SG90Servo.pdf

Morgan, J. (2014). A Simple Explanation Of 'The Internet Of Things'. Verkregen van https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#503ce8cd1d09

Nordic-semiconductor. (2017). nRF Connect mobile application. Verkregen van https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Connect-for-mobile-previously-called-nRF-Master-Control-Panel

Onion. (2017). Omega2: 5 dollar Linux Computer with Wi-Fi, Made for IoT. Verkregen van https://www.kickstarter.com/projects/onion/omega2-5-iot-computer-with-wi-fi-powered-by-linux

Raspberry. (2016). Raspberry Pi 3 Model B. Verkregen van https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

RaspberryPi. (2015). Dimensions of Raspberry Pi 3B. Verkregen van https://www.raspberrypi.org/documentation/hardware/raspberrypi/mechanical/RPI-3B-V1_2.pdf

raspberrywiki. (2016). RPI Lihium Expansion Board. Verkregen van http://www.raspberrypiwiki.com/index.php/RPI_Lithium_Battery_Expansion_Board_SKU:435230

Raspbian. (g.d.). Raspbian. Verkregen van https://www.raspbian.org/

StudentEnMobiliteit. (2017). Een fiets huren bij studentenmobiliteit. Verkregen van http://www.studentenmobiliteit.be/student/nl/een-fiets-huren.php

Uber. (2016). Hoe werkt Uber. Verkregen van https://help.uber.com/h/738d1ff7-5fe0-4383-b34c-4a2480efd71e

VoCore. (2017). VoCore 2 Ultimate specifications. Verkregen van http://vocore.io/v2u.html

Lijst van figuren