Afbeelding met tekst, illustratie

Automatisch gegenereerde beschrijving

Gebroeders de Smetstraat 1, 9000 GENT

Verslag auto ont- en vergrendelen via bluetooth

Michiel Decrock

Mathis Delbeke

Jarno Lannoo

Professionele Bachelor Elektronica-ICT / Fase 2

Academisch jaar: 2021-2022 Opdrachtgever: dr. Piet Coussens

Inhoudsopgave

[Inleiding 1](#_Toc99909734)

[Project overview 2](#_Toc99909735)

[1 Introductie STM32 Nucleo 3](#_Toc99909736)

[2 CAN-hacker 4](#_Toc99909737)

[2.1 Nood aan CAN-hacker 4](#_Toc99909738)

[2.2 Gebruikt CAN-sniffer 4](#_Toc99909739)

[2.3 Techniek van sniffen 5](#_Toc99909740)

[3 CAN-bus via Nucleo 6](#_Toc99909741)

[3.1 Schema verbinding CAN-bus met Nucleo 6](#_Toc99909742)

[3.2 Programmacode unlocken & locken van de deuren. 7](#_Toc99909743)

[4 Bluetooth via HC-06 9](#_Toc99909744)

[4.1 Ontvangen van karakters 9](#_Toc99909745)

[4.2 Filteren van berichten 9](#_Toc99909746)

[4.3 Programmatie 10](#_Toc99909747)

[4.4 Gebruik 12](#_Toc99909748)

[4.5 Interface HC-06 12](#_Toc99909749)

[4.6 Beveiliging 12](#_Toc99909750)

[5 Totale werking 14](#_Toc99909751)

[5.1 Hardwareschakeling 14](#_Toc99909752)

[5.2 Programmatie 15](#_Toc99909753)

[Bronnenlijst 16](#_Toc99909754)

# Inleiding

Tegenwoordig hebben moderne auto’s een keyless-entry systeem. Wanneer de sleutel dicht in de buurt komt van de bijhorende auto, zal deze automatisch openen, zonder dat de gebruiker naar zijn sleutel moet zoeken.

In dit project is het doel om een auto te ont- en vergrendelen via een smartphone. Zo kunnen oudere auto’s ook een keyless-entry systeem bekomen. Uiteraard is er wel steeds een fysieke sleutel nodig om de auto daadwerkelijk te starten. Hiervoor zijn twee mogelijke oplossingen. De auto kan automatisch ontgrendelen of vergrendelen wanneer de smartphone van de gebruiker dicht in de buurt komt. Ook is het mogelijk om een smartphone-applicatie te gebruiken die enkele knoppen bevat, waarmee de auto kan sluiten of openen. Wegens tijdsgebrek bevat dit project de tweede oplossing. Als uitbreiding zou er voor geopteerd kunnen worden om dit toch automatisch te laten gebeuren.

Het plan van aanpak start bij het onderzoeken hoe een auto zijn data verstuurt. Hierbij zal ook gekeken worden naar de data die over de CAN-bus beweegt, wanneer de auto gesloten of geopend wordt. Daarna zal onderzocht worden hoe we met onze microcontroller dezelfde data op eigen initiatief kunnen versturen over de CAN-bus. Eenmaal dit gebeurt is kan er nagedacht worden over een bluetooth verbinding tussen een mobiel device en microcontroller. Tot slot zal alles samen moeten werken voor één geheel te vormen. Dit betekend dat bij het gebruik van de smartphone er via bluetooth verstuurd wordt wat er moet gebeuren (sluiten of openen). Vervolgens zal de microcontroller de nodige data over de CAN-bus sturen om de auto te ont- en vergrendelen.

Het verslag start met het kort schetsen van het project zodat er een duidelijk beeld ontstaat. In een tweede hoofdstuk zal het gebruikte Nucleo bordje besproken worden. Daarna wordt aangetoond hoe de nodige CAN-frames om de auto te ontgrendelen en vergrendelen gevonden zijn. Vervolgens wordt besproken hoe we de gevonden data zelf kunnen sturen over de CAN-bus. Daaropvolgend een hoofdstuk over de bluetoothconnectie tussen smartphone en HC-06. Als laatste hoofdstuk wordt de totale werking van het project aan de hand van code uitgelegd.

# Project overview

Onderstaande afbeelding toont de verschillende takken van dit project aan. Het project start bij het STM32 Nucleo bord, het centrale punt.   
  
Het STM32 Nucleo bord zal input ontvangen van de smartphone via bluetooth en op basis daarvan de nodige data op de CAN-bus sturen. Omdat dit bordje geen ingebouwde bluetooth bevat, is er een externe module genaamd HC-06 vereist waarmee er serieel gecommuniceerd wordt.   
  
Het STM32 Nucleo bord bevat wel CAN onboard, echter kan dit bordje niet rechtsreeks aan de CAN-bus aangesloten worden. Hiervoor is er een TJA1050 of dus CAN-tranceiver voor nodig. Dit zijn de nodige hardware componenten, die vereist zijn voor dit project te laten slagen.

Afbeelding met tekst, schermafbeelding

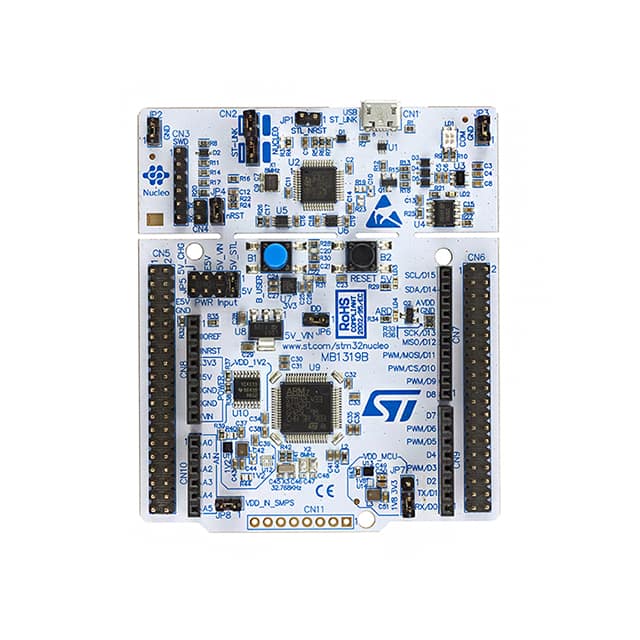
Automatisch gegenereerde beschrijving

Figuur : Project onderdelen

Dit project is specifiek ontwikkeld voor een smartphone dat Android runt en een Ford Fiesta uit 2012. Uiteraard kan ieder bluetooth device verbinden met de HC-06, alleen is de applicatie voor het openen en sluiten van de auto enkel voor Android smartphones. Dit kan gemakkelijk uitgebreid worden door een extra applicatie voor de (betere) IOS-gebruikers.   
  
Elke autofabrikant kan zijn eigen data kiezen die er specifiek voor zorgt dat de auto zal sluiten of openen. Hierdoor kan dit project niet onmiddellijk gebruikt worden voor iedere auto. De nodige CAN-frames om de auto te laten openen of sluiten is hard coded in het programma die op het STM32 Nucleo bordje runt. Door in slechts vier lijntjes code enkele gegevens te veranderen, zou er een andere auto kunnen gebruikt worden.

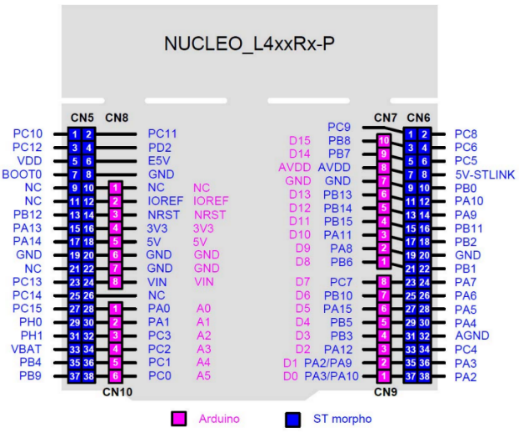
# Introductie STM32 Nucleo

Het STM32 Nucleo L433RC-P bord biedt gebruikers een betaalbare en flexibele manier om nieuwe concepten uit te proberen en prototypes te bouwen met een STM32-microcontroller. Deze heeft gelijkaardige features als de bekende Arduino-borden.



Figuur : Overzicht STM32 Nucleo L433RC-P

Onderstaande Figuur 3 toont de pinout van het bordje. De STM32 Nucleo L433RC-P biedt meer features als een Arduino Uno en dat terwijl deze voor een lagere prijs wordt verkocht. Enkele van deze interessante features: 52 fast I/O pins, ingebouwde OPAMP, twee comparatoren, ARM® 32-bit Cortex®-M4 CPU met FPU, True random number generator, CRC calculation unit…



Figuur : Pinout L433RC-P

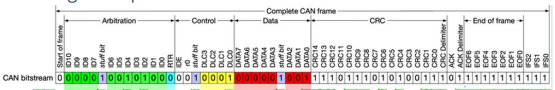
Voor het datacommunicatie project is het ook interessant dat dit bordje meerdere protocollen ondersteund. Zo heeft deze USB 2.0, I2C, USART, SPI en CAN. Vooral het laatste is zeer handig voor dit project. Dit uiteraard omdat alle instructies in een auto via CAN worden verstuurd.

Het bordje met een STM32L433 microcontroller wordt geproduceerd door het bedrijf STMicroelectronics. Deze heeft ook de STM32 Cube IDE ontwikkeld, een programmeeromgeving voor microcontrollers van deze familie te programmeren in C of C++. Deze bevat een compiler, uploader als ook een debugger. Ook kan er voor gekozen worden om het bordje te programmeren via ST Mbed. Dit is ook een IDE ontwikkeld door het bedrijf ST, maar bevat een actieve community en is beter gedocumenteerd. Mbed bevat ook handige bibliotheken waarbij er achterliggende code reeds geschreven is.

# CAN-hacker

## Nood aan CAN-hacker

Zoals eerder vermeld zal iedere autofabrikant zijn eigen data kiezen, die ervoor zorgt dat de auto zal sluiten of openen. Aangezien het CAN-protocol gebruikt wordt, zijn er een aantal parameters die gekozen kunnen worden.



Figuur 4: CAN data link layer

Het is belangrijk dat de indentifier, die uit drie hexadecimale cijfers bestaat, bekend is. Ook moet het aantal bytes en de daadwerkelijke data geweten zijn. Als deze parameters heruitgestuurd worden over de CAN-bus, zal de auto vergrendelen of ontgrendelen.  
  
Om de hierboven besproken parameters te vinden, is er een CAN-sniffer vereist. Autofabrikanten zullen zelf hun CAN-frames kiezen en deze ook niet zomaar vrijgeven. Zo zijn er een aantal gratis CAN-sniffers online te vinden.

## Gebruikt CAN-sniffer

Onderstaande figuur bevat een screenshot van de gebruikte software genaamd CANHacker. Hiervoor wordt een MCP2515 als interface gebruikt. Deze zal via een OBDII connector aan de CAN-bus aangesloten worden en zal via SPI de data die over de CAN-bus beweegt, doorsturen naar een Arduino-bordje. Indien de bijgeleverde Arduino-code runt, kan de UI opgestart worden. Daar wordt er eerst geconnecteerd en vervolgens gestart met sniffen. Indien de knop op de fysieke autosleutel ingedrukt wordt, zal er veel CAN-frames of op het scherm verschijnen.

Afbeelding met tafel

Automatisch gegenereerde beschrijvingAfbeelding met tekst, krant, document

Automatisch gegenereerde beschrijving

Figuur 5: CANHacker V2 gratis software

## Techniek van sniffen

Eén of meerdere van de binnengekomen regels bevat het nodige om het commando (sluiten of openen) te herhalen. Gelukkig bevat de CANHacker software ook een functie om geselecteerde regels heruit te zenden op de CAN-bus.   
  
Hierbij is de beste techniek om de eerste helft van de binnengekomen regels te verwijderen en de rest te laten afspelen. Als de auto ont- of vergrendeld betekend dit dat de nodige data nog steeds aanwezig is. Vervolgens herhaal je dit tot dat de auto niet meer opent of sluit, dan weet u dat de nodige data verwijdert is. Zo kan je veel irrelevante regels wegwerken en testen met een beperkt aantal regels totdat enkel het nodige overblijft.  
  
In onderstaande tabel staat de gevonden data voor het openen en sluiten van de auto (Ford Fiesta 2012). Het eerste getal is niet zo relevant, dit is een timestamp van elke opgevangen regel. Vervolgens wordt de indentifier, die uit drie hexadecimale cijfers bestaat, meegegeven. Daarna vier bits voor het beschrijven dan de data lengte, deze is steeds 8. Ten slotte de 8 databytes die vereist zijn om de auto te laten sluiten of openen.

Tabel : Gevonden data voor openen of sluiten

|  |  |
| --- | --- |
| CLOSE ALL DOORS | Afbeelding met tekst  Automatisch gegenereerde beschrijving |
| OPEN ALL DOORS |  |

In bovenstaande tabel is op te merken dat voor het openen van de deuren slechts één regel nodig is, en voor het sluiten drie. Als één van deze drie regels niet uitgestuurd wordt over de CAN-bus, zal de auto niet sluiten.

# CAN-bus via Nucleo

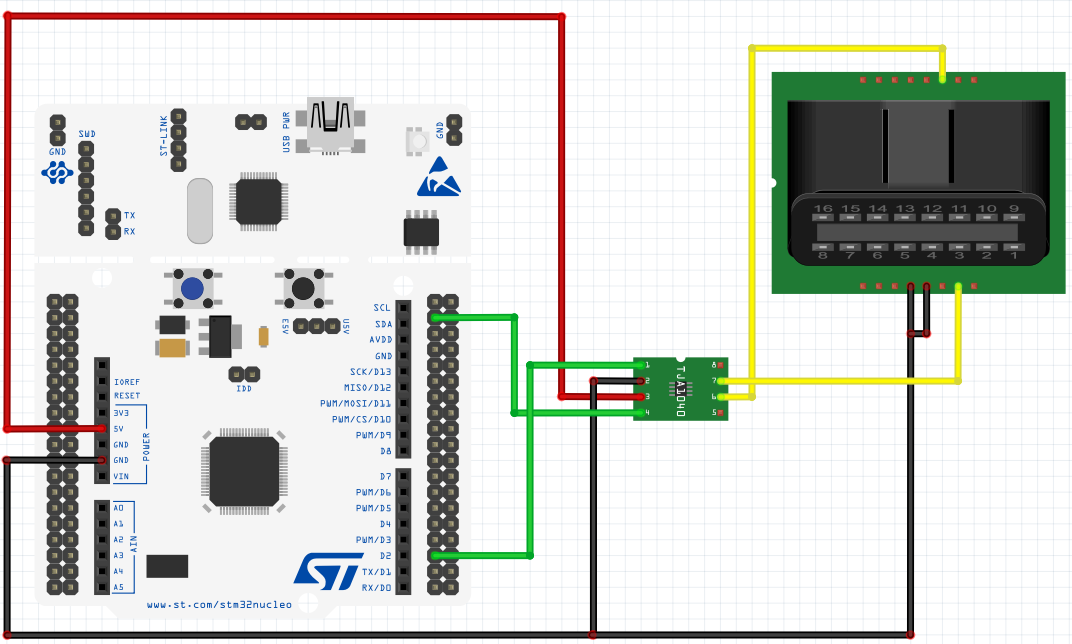
Een rechtstreekse verbinding tussen een Nucleo bordje en een OBDII-connector is niet mogelijk. Het probleem hierbij is dat een Nucleobordje zelf niet een hoog of laag niveau van de CAN-bus kan genereren. Een oplossing hiervoor is door gebruik te maken van een transceiver, meer bepaald de TJA1050 zoals onderstaande figuur 6.

Tabel 2: Pin-out transceiver TJA1050 & OBDII-connector

|  |  |
| --- | --- |
| **Transceiver: TJA1050** | **OBDII-connector** |
| Figuur : Overzicht TJA1050 met pin-out | Afbeelding met tafel  Automatisch gegenereerde beschrijving  Figuur 7: OBDII-connector met pin-out |

## Schema verbinding CAN-bus met Nucleo

Onderstaande Figuur 8 toont de aansluitingen tussen het Nucleobordje, de transceiver en de OBDII-connector.



Figuur : Aansluitschema CAN-bus met Nucleo

Tabel : Aansluitingen Nucleobordje met OBDII-connector

|  |  |
| --- | --- |
| **Aansluitingen** | |
| **TJA1040 = TJA1050 in pin-out** | **OBDII-connector** |
| * GND -> GND * VCC -> 5 V * TXD -> PA\_12 (Nucleo) * RXD -> PB\_8 (Nucleo) | * Pin 4 + 5 -> GND * Pin 11 -> CANH (TJA1050) * Pin 3 -> CANL (TJA1050) |

## Programmacode unlocken & locken van de deuren.

Figuren 9 & 10 zijn methodes uit het bestand main.cpp die zich op GitHub bevindt. De methode “lockDoors” op Figuur 9 toont de programmatie in verband met het sluiten van de auto. Vervolgens geeft Figuur 10 de code voor het openen weer van het voertuig.

Voor het sluiten van de auto moeten er drie pakketten worden verstuurt. Binnen de for-lus, wordt er een object aangemaakt met de naam “CANMessage”. Hierin wordt een id-nummer en acht databytes bewaard. Binnenin de do-while-lus blijft de variabele “transaction” false zolang dat het pakket niet succesvol ontvangen is op de bus. Dit proces herhaald zich nog twee maal om de overige pakketten te verzenden.



Figuur : Programma-methode unlockDoors

De code voor het openen van het voertuig is eenvoudiger. Dit doordat er maar één pakket moet worden verstuurt. Het maken van een object met een id-nummer en de acht databytes is hierbij ook noodzakkelijk. Tot slot wordt het pakket continu verstuurt totdat er een bevesting komt dat het ontvangen is.



Figuur : Programma-methode lockDoors

# Bluetooth via HC-06

## Ontvangen van karakters

Om data van de smartphoneapp te kunnen ontvangen, moeten de binnenkomende karakters worden verwerkt. Om dit te doen biedt de gebruikte Nucleo enkele opties:

* Blocking mode
* Interrupt mode
* Direct Memory Access (DMA)

De beste optie is DMA, aangezien de processor hier niet nodig is tijdens een transactie. De data wordt “gestreamd” naar een gereserveerde geheugenplaats. Echter kan dit enkel indien de berichten die binnenkomen een vaste lengte hebben. In dit project is dit niet het geval, dus deze methode is niet bruikbaar.

Voor dit project wordt interrupt mode verkozen. Deze methode werkt door na elk ontvangen karakter een interrupt af te vuren. De processor gaat vervolgens het binnenkomende karakter gaan verwerken. Deze methode is meer CPU-intensief, maar werkt optimaal tot snelheden van 115200 bauds per seconde (bps).

## Filteren van berichten

Omdat niet elk binnenkomend bericht nuttig is, is er nood aan een filter. Dit voorkomt dat statusberichten van de module zelf genegeerd worden. Ook is er nood aan een soort stopkarakter, die aangeeft waar het einde van een bericht is.

Om aan bovenstaande vereisten te voldoen, programmeren we in de app twee verschillende “delimiters”. Dit zijn karakters die, bij het binnenkomen, een actie in gang zetten. Het procent karakter geeft het begin van een bericht aan, het andere karakter geeft het einde aan.

Het programma wordt zodanig opgesteld dat indien er een ‘%’ karakter binnenkomt, de microcontroller begint op te letten wat er binnenkomt. Zo wordt een buffer aangemaakt om alle volgende karakters in op te slaan. Ook wordt een “pointer” aangemaakt. Deze zal bijhouden waar het volgende karakter in de buffer moet geplaatst worden.

Vervolgens komen er andere tekens binnen, die samen het commando uit de app opstellen. Elk karakter wordt opgeslagen in de eerder vermelde buffer.

Eens het bericht op z’n einde is, komt er een regeleindekarakter binnen. Dit is de tweede “delimiter”. Deze geeft aan dat het bericht ten einde is. Ook wordt een vlag ingesteld om het programma te laten weten dat er een transactie met een commando heeft plaatsgevonden.

## Programmatie

Om de eerder beschreven werken te realiseren, moeten we alles omzetten naar functionele c++ code. Dit doen we in MBED studio, een IDE van ARM.

Eerst moet de seriële interface van de chip geactiveerd worden. Dit wordt zowel voor USB als voor Bluetooth gedaan. Dit laat toe om te debuggen via USB. Dit kan gezien worden in Figuur 11.



Figuur : Code om seriële interface te initialiseren

Vervolgens moet er een buffer voor de karakters en een pointer worden bijgehouden in het geheugen. Dit kan gezien worden in volgende figuur (Figuur 12).



Figuur : Bijhouden van buffer en plaats in buffer

Om aan een inkomend karakter een interrupt te koppelen, wordt volgende code gebruikt (Figuur 13).



Figuur : Interrupt koppelen

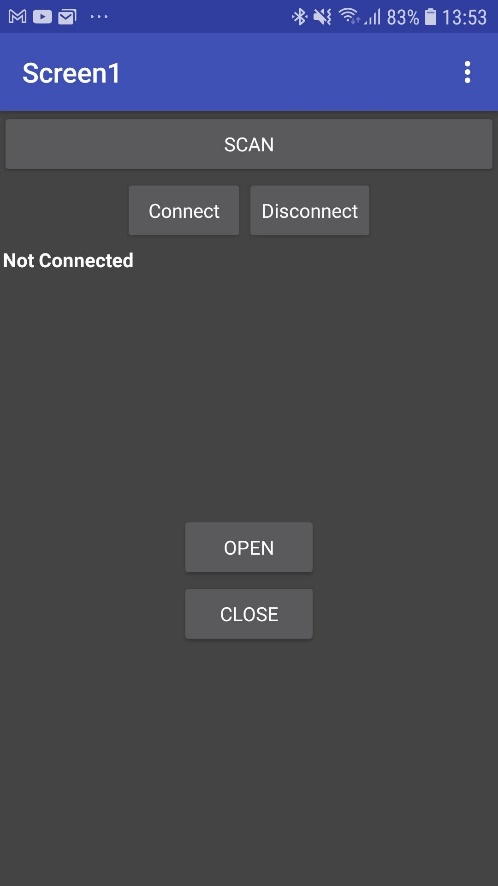
Vervolgens schrijven we de interruptroutine om de ontvangen karakters te verwerken. Ook hier is elke regel van commentaar voorzien om de werking duidelijk te maken. Deze routine is te zien in volgende afbeelding (Figuur 14).



Figuur : Interruptroutine

## Gebruik

Om de auto te kunnen bedienen, is een app vereist. Deze app bevat de nodige commando’s waarop de nucleo reageert.

Deze app kijkt naar de lijst met gekoppelde apparaten op de smartphone, en maakt een seriele connectie als we op “connect” klikken. Nu kunnen we met twee knoppen de auto openen en sluiten. Volgende afbeelding (Figuur 15) toont hoe deze app eruit ziet.

Figuur : smartphoneapplicatie

## Interface HC-06

De communicatie met de smartphone verloopt via een HC-06 Bluetoothmodule. Deze module heeft een seriële communicatie interface onder de vorm van UART. Om data te versturen en te ontvangen is er dus nood aan een UART op onze microcontroller. Poort 1 wordt hiervoor verkozen, aangezien poort 2 gebruikt wordt voor USB-communicatie met de PC.

## Beveiliging

Om te voorkomen dat iedereen de auto kan openen, moet er een beveiliging aanwezig zijn. In het geval van de HC-06 is dit een 4-cijferige pincode die moet worden ingegeven in de smartphone alvorens te connecteren.

Afbeelding met tekst

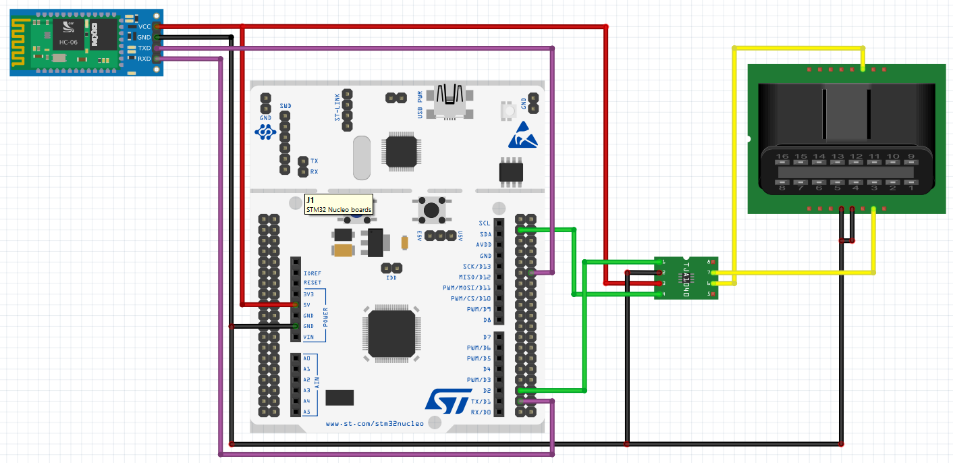
Automatisch gegenereerde beschrijvingVerder is een pakketstructuur nodig die ervoor zorgt dat we tijdens transmissie geen data verliezen. De Bluetoothstandaard heeft zo’n pakketstructuur standaard ingebakken, dus dit moet niet zelf voorzien worden. Volgende figuur toont deze pakketstructuur aan.

Figuur : Bluetoothpakketstructuur

# Totale werking

## Hardwareschakeling

Volgende figuur (Figuur 17) toont de schakeling die zich binnenin de auto bevindt. Links staat de HC-06 Bluetooth module afgebeeld. In het midden staat de Nucleo, en rechts een CANbus-transceiver verbonden met een OBDII connector.



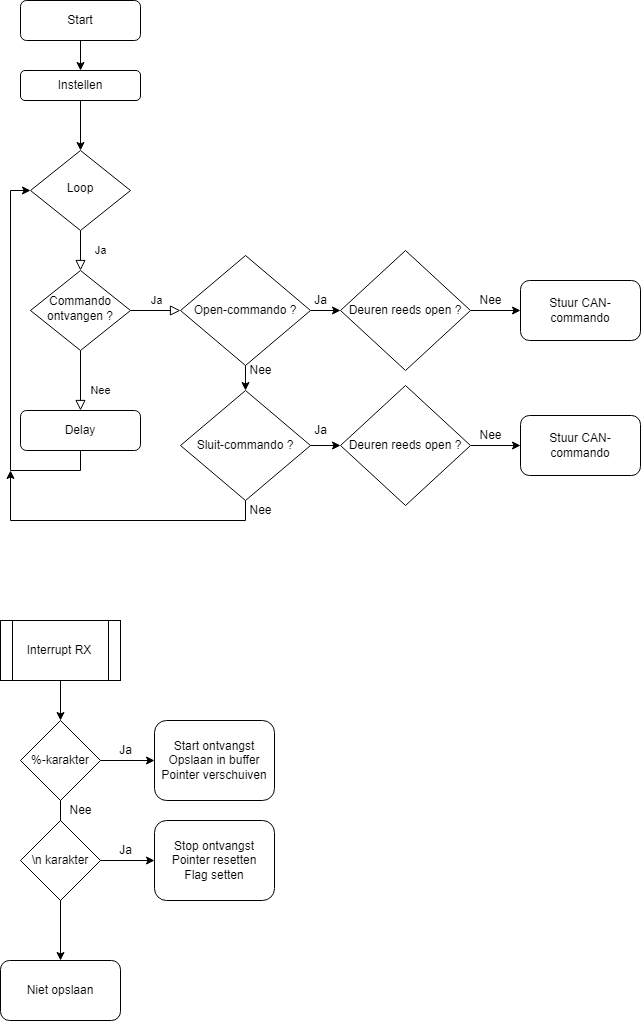
Figuur 17: Hardwareschakeling

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Figuur 18: Foto hardwareschakeling

## Programmatie

Het volledige c++ programma is te vinden in de Git-repo van dit project. Op de volgende afbeelding (Figuur 18) staat wel een overzichtelijke flowchart die aangeeft hoe het programma in elkaar zit.

Figuur : Flowchart van het programma

# Bronnenlijst

**CAN Hacker**

https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/709/HC-05\_ATCommandSet.pdf

**Stm32 API**

<https://www.st.com/resource/en/user_manual/dm00173145-description-of-stm32l4l4-hal-and-lowlayer-drivers-stmicroelectronics.pdf>

**HC-06 AT command set**

<https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/709/HC-06_ATCommandSet.pdf>

**Git repo**

<https://gitlab.com/ikdoeict/mathis.delbeke/keylesscaropening>