Raven

H5 – Projekt opgave

Rasmus W. Knudsen & Niklas W. Micheelsen

https://github.com/nikhunter/Raven

Indholdsfortegnelse

Forord

Denne rapport skrevet ud fra vores projekt under H5, den er skrevet af Niklas W. Micheelsen & Rasmus W. Knudsen. Rapporten omhandler en prototype af vores 'Vogn monitorering og analyse' softwarepakke.

Vores projekt er tænkt som at være en prototype af en såkaldt "black box" som kan sælges som en del af en serviceaftale til større transport firmaer, såsom DHL, UPS, 3x34 osv.

Vores Development prototype er delt op i tre dele – OBD-II connector, Arduino og en Android mobil. Når vores "proof of concept" er fuldt fungerende og Raven-Desktop er færdigudviklet kan vi begynde på fase 2 af Raven.

I fase 2 vil vi arbejde på at minimere prototypen til en lille kasse som man vil kunne efterlade tilsluttet i bilens OBD-II stik. Med en indbygget GPS og et 3G GSM modem vil den kunne opdatere og sende data til Raven servers.

Navnet Raven har vi valgt som en reference til Odins to ravne, Hugin og Munin, som overvåger landskabet fra luften.

Indledning

Raven GPS er en suite af software designet for at give transportselvskaber muligheden for at tracke og analysere sine chauffører, vogne og deres ruter.

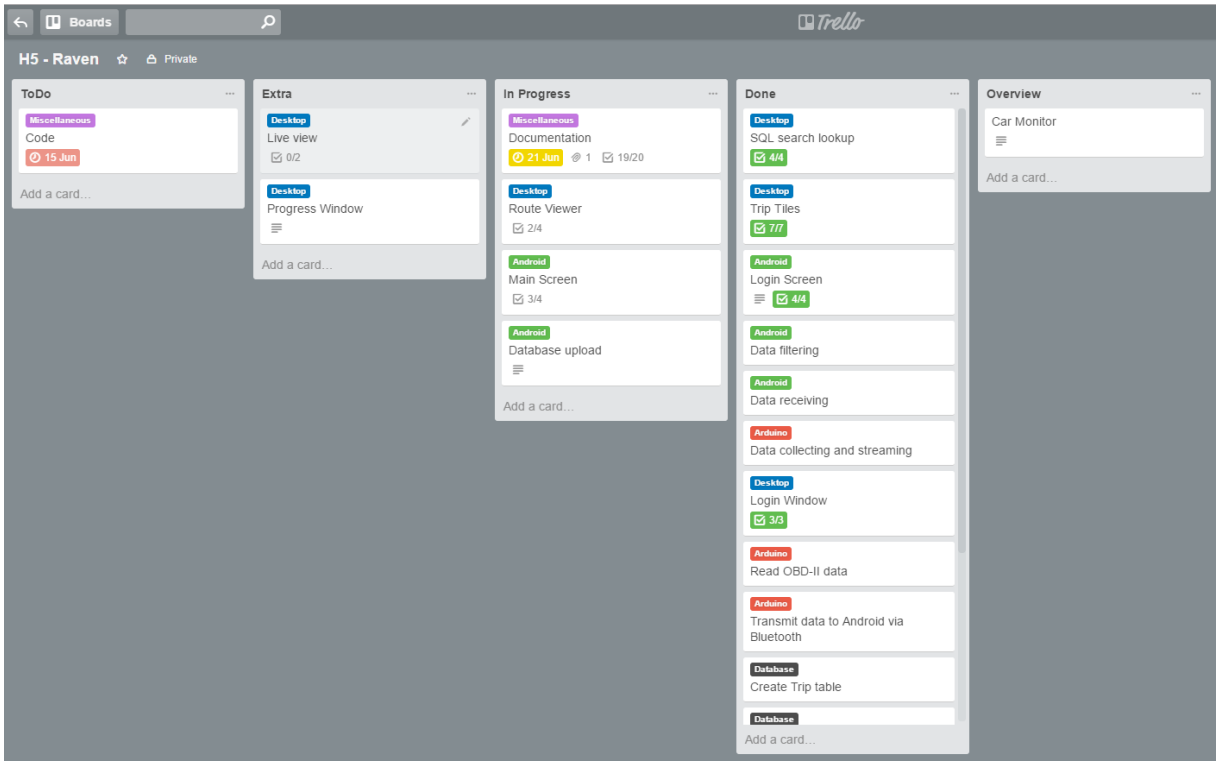
Dette bliver udført ved hjælp af bilens OBD-II stik, der er en ny standard inden for Auto industrien. OBD står for "On-Board Diagnostics" og giver adgang til bilens CAN Bus som giver adgang til 'hjernen' af bilen dvs. data så som hastighed, RPM, Motor stress og meget mere. Disse informationer bliver sendt til chaufførens smartphone over Bluetooth, der derefter vil blive sendt til Raven GPS' database.

Et par eksempler på i hvilken situation dette vil være brugbart er f.eks. En arbejdsgiver får en klage over en medarbejders vanvidskørsel. Der vil arbejdsgiver kunne åbne Raven programmet, skrive medarbejderens nummerplade, stelnummer eller potentielt andre identificerende detaljer og se alle ruter en specifik medarbejder har foretaget. Arbejdsgiver vil så kunne trykke på en given rute og verificere at der er hold i klagen og tage aktion.

Eller at en af firmaets vogne brød sammen eller nedkøling af læsset ikke fungerer ordentligt så kan firmaet finde ud af hvor den er ved at slå den op og tilkalde den nærmeste vogn som er tilgængelig til dens position.

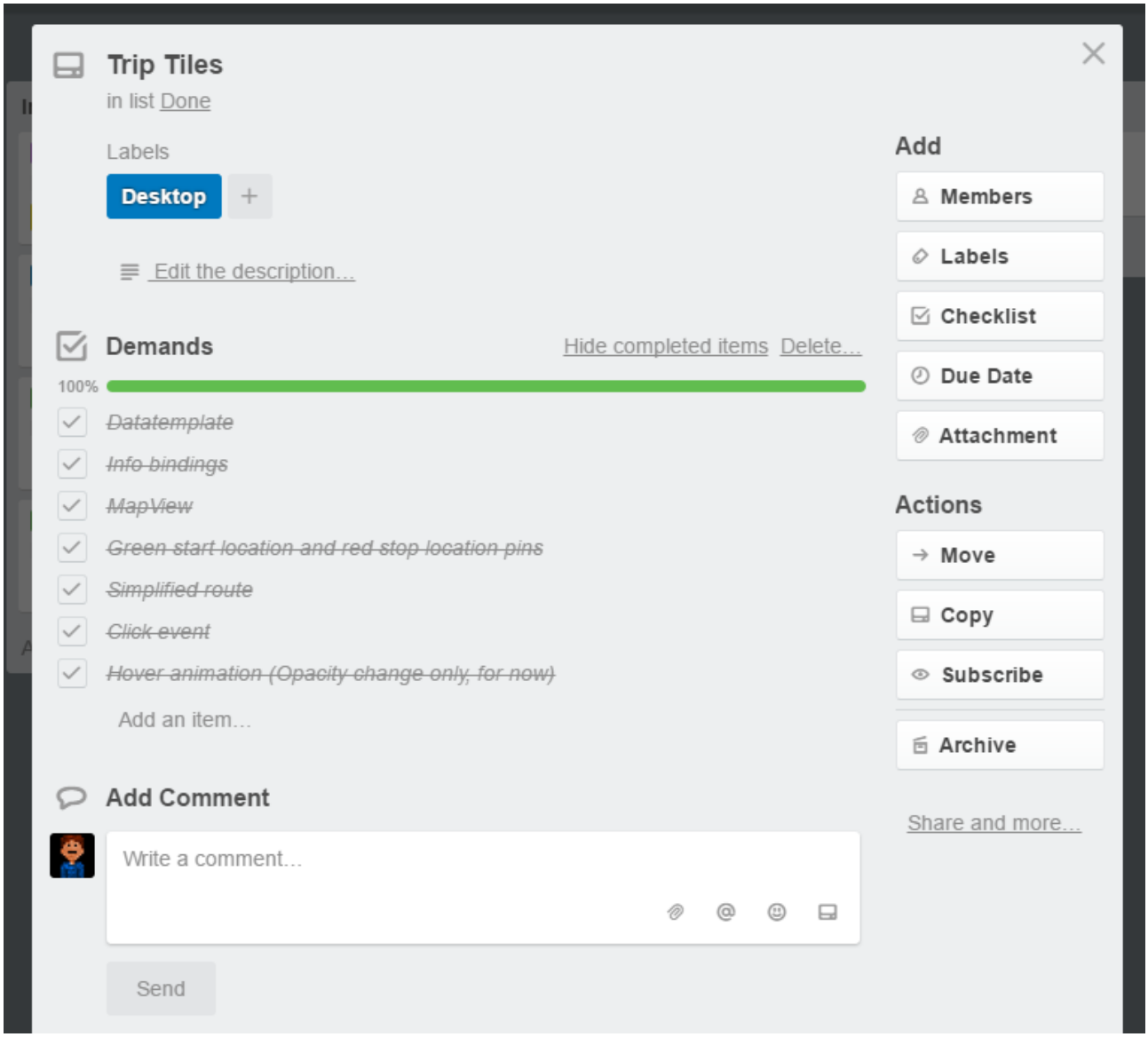
Projektstyring

Til projektstyring har vi brugt Kanbanboard og servicen "Trello", Trello giver os mulighed for hurtigt at kunne oprette nye opgraver med detaljeret beskrivelser. Trello virker på den måde at man kan organisere opgaver som digitale post-it notes og tilføje dem til forskellige kolonner.



(fig. 1)

Når en ny opgave skal oprettes starter den ved at blive tilføjet til en af vores 'To-do' kolonner for vores forskellige projekt dele. Hvis denne opgave angiver en opgave for Raven-Arduino vil den blive markeret som en Arduino opgave og tilføjes til vores 'To-Do' kolonne, når opgaven bliver begyndt på bliver opgaven flyttet til 'In-progress' og eventuelle under opgaver vil derefter blive tjekket af på listen en for en. Når alle under opgaver er fuldført vil opgaven bliver flyttet en sidste gang, til vores 'Done' kolonne der over tid vil blive længere og længere.



(fig. 2)

Problemstilling

Hvordan kan det være, at vi har valgt at lave dette produkt?

Det er hovedsageligt fordi at vi har stor interesse inden for biler, men også fordi at det ville være spændende at lave et projekt som både var afhængelig af hardware og som

benyttede flere platforme, noget som ligner et realistisk produkt og som inkluderede et område vi ikke havde særligt meget kendskab til.

Hvor kom ideen fra?

Ideen kom fra at en af vores fædre som havde lavet noget lignede for mange år siden. Noget som han forslog kunne erstattes hvis nogle tog konceptet og forbedrede det. Realistisk set så er hvad vi har lavet en prototype da for at det kan erstatte den gamle version mangler der meget funktionalitet. Men det er hovedsageligt der inspirationen kom fra.

Hvad ville vi gerne lære igennem arbejdet?

Vi ville gerne lære mere om RTOS da det er et ekstremt brugbart værktøj i den virkelige verden, det har været udfordrende at arbejde med Arduino.

Problemformulering

Problemer, bekymringer eller ting der bør overvejes inden for denne type af produkt.

Hvordan vil vi overføre data fra Arduino til Android?

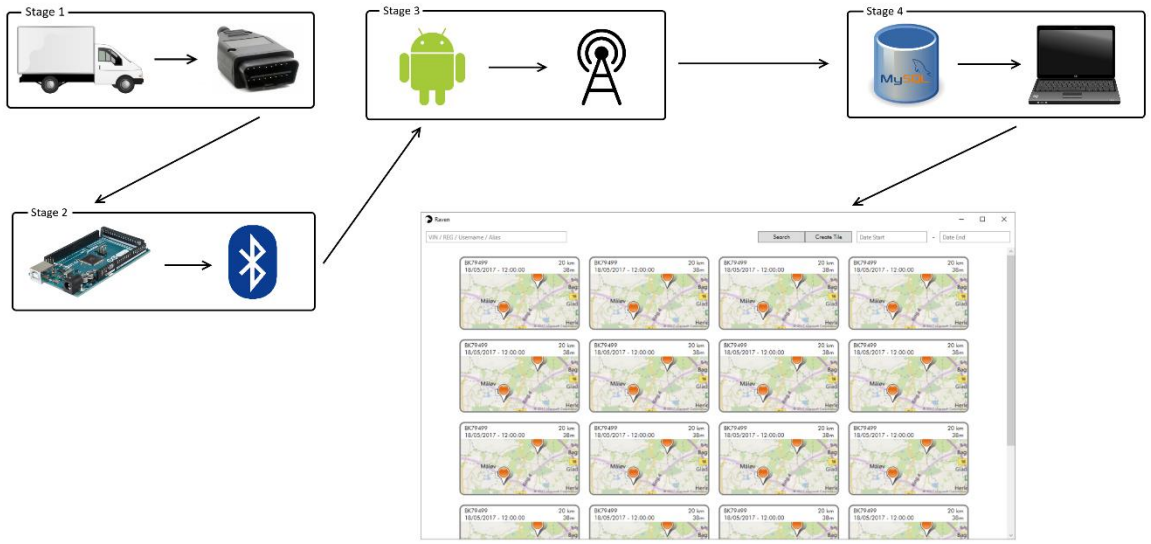
Til vores prototype version valgte vi at bruge Bluetooth imellem Arduino og Android da et Bluetooth modul til Arduino ikke koster mange penge.

Vi havde overvejet et GSM modul (data forbindelse via SIM kort), så var der ikke behov for Android, men vi syntes at et GSM modul ville være for dyrt i første omgang.

Hvordan vil vi lagre data?

Vi valgte at lagre vores data i Json format på en SQL server, det endte med at blive en LONGTEXT da det største antal tegn tilladt. Der er sikkert hunderede vis af bedre muligheder, men da det er relativt hurtigt at lave indsæt eller udtræk og at vi indtil videre ikke har ramt grænsen for vores logs.

Rigt billede



(fig. 3)

Krav

Desktop:

- Søgefunktion hvor brugere kan filtrere ruter efter registrerings numre, førerens brugernavn og andre parametre.
- Vise en boks for hver tur kørt, som indeholder generel information og et kort over ruten. Information som, registrerings nummer, distance, varighed, dato og tid på starten af turen.
- Detaljeret tur visning som viser den fulde rute, punkter som er blevet registreret i løbet af turen som kan vise dato, tid, hastighed, omdrejninger i minuttet, breddegrad og længdegrad på det tidspunkt hvis brugeren klikker på et af punkterne.

Android:

- Modtag data fra Bluetooth modulet på Arduino'en.
- Filtrere modtaget Json linjer og formatere til gyldig og læseligt Json logs.
- Send filtreret data til SQL via GSM/Data forbindelse på Android.

Arduino:

- Læs fra OBD-II.
- Efter en læse cyklus start sending.
- Send data til Android i Json format over Bluetooth til Android.

Database:

- Skal have et login table med registreringsnummer, brugernavn og krypteret password.

- Skal have et trip table hvor vi kan gemme ture med start/slut tidspunkt, førerens brugernavn, førerens registrerings nummer og selve loggen.

Udviklingsmiljøer

Raven-Desktop

Miljø: Visual Studio 2017
Raven-Desktop er skrevet i WPF (C#, XAML) ved brug af Bing Maps SDK

Raven-Arduino

Miljø: Arduino IDE
Raven-Arduino er skrevet i C.

Raven-Android

Miljø: Android Studio
Raven-Android er skrevet i java.

Værktøjer

Balsamiq Mockups 3

Visual Studio

Android Studio

Arduino IDE

MYSQL

GitHub

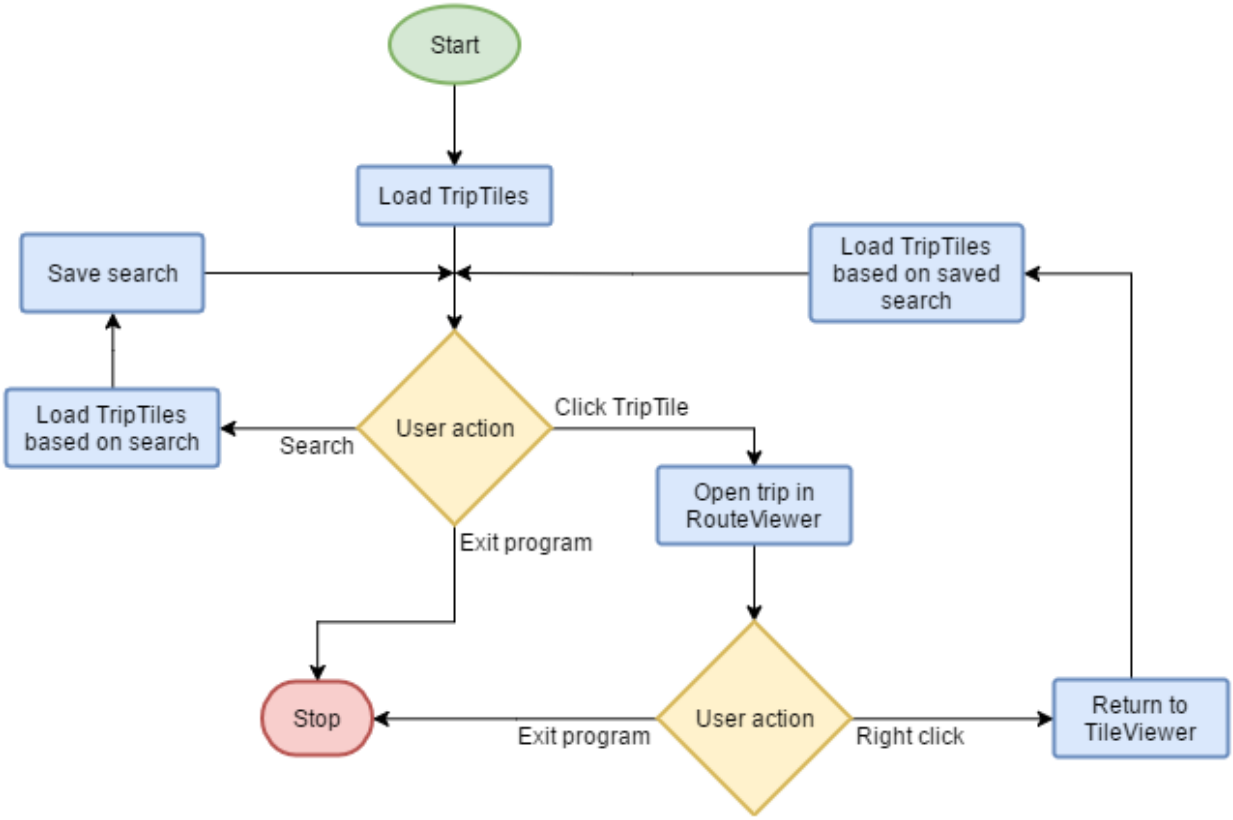
Trello

Flowcharts

Et flowchart diagram er en oversigt af et helt eller dele af et programs "flow".

Et flowchart diagram er en effektiv måde at forstå de trin eller retninger programmet bevæger sig i. Det er specielt effektivt i at forklare hvordan et program fungerer også hvis personen man viser det til ikke er en udvikler.

Raven-Desktop

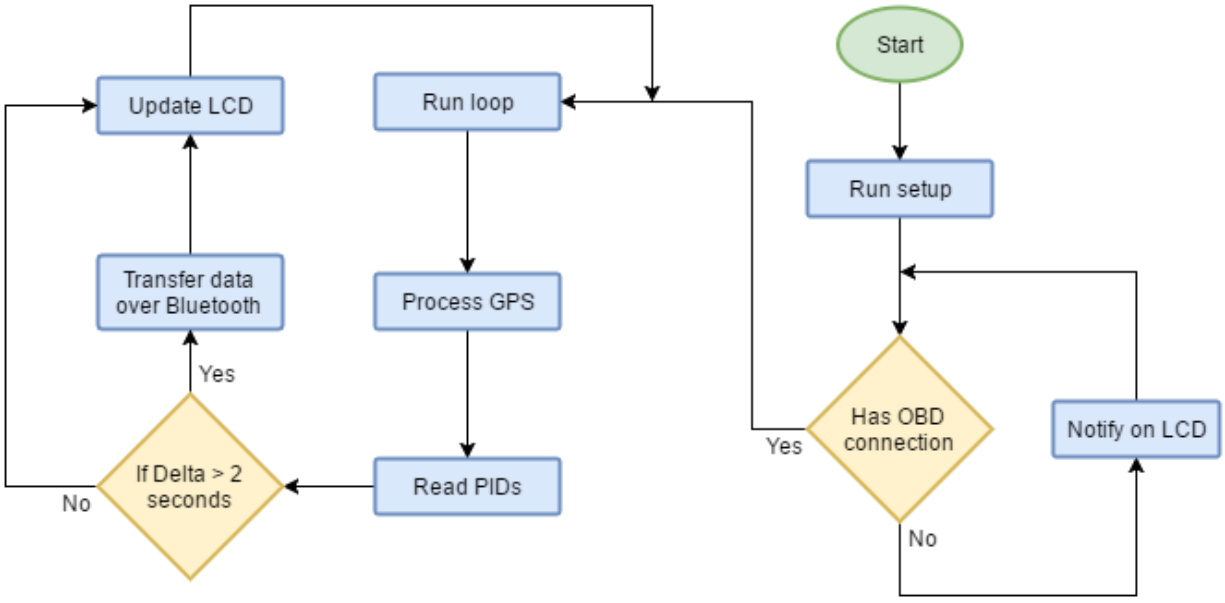


(fig. 4)

- TripTiles er bokse der viser general information omkring en tur, sammen med et kort af den kørte rute.

Eksempel af TripTiles kan ses under afsnittet Kodebeskrivelse ved MainWindow(1.1.0)

Raven-Arduino



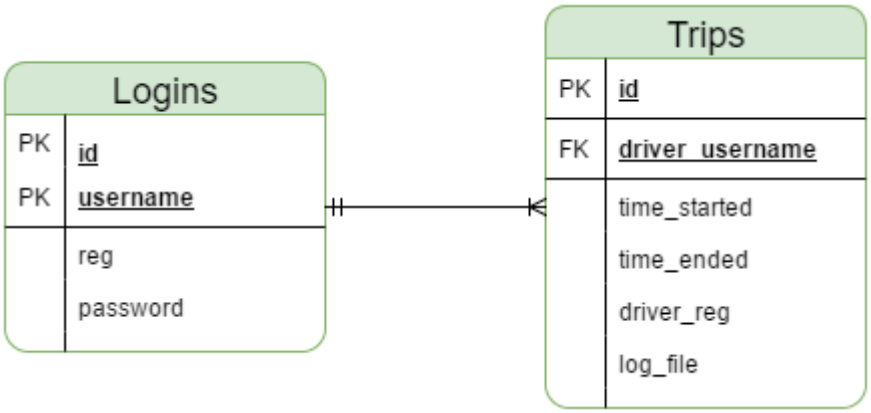
(fig.5)

- Der findes ikke en 'terminator' i vores flowchart da vores Arduino looper konstant indtil strømmen afbyddes på OBD.
- Read PIDs – Læser informationer fra bilens computer via OBD og forbereder dem i en JSON string.

ER-Diagram

Billedet herunder forstiller vores database opsætning. Originalt tænkt til kun at være en midlertidig database indtil vi fandt en bedre opsætning. Dette skulle vise sig at være en fejl, da vi hurtigt blev låst fast i vores SQL opsætning efterhånden som der blev skrevet diverse scripts og insert statements.

Vores SQL er sat sammen så simpelt som muligt. Efterhånden som der kom mere og mere data blev vi enige i at der ikke var nogen grund til at gøre det unødigt kompleks.



(fig. 6)

Use Cases

Raven-Desktop

Titel	Søgning af ture
Formål	At finde ture som matcher et id, registrerings nummer eller brugernavn
Aktører	Bruger
Frekvens	Hver gang aktøren trykker på 'Search' knappen
Startbetingelser	Aktøren er logget ind og hoved vinduet er kommet frem
Beskrivelse	1. (Valgfrit) Indtast søgeord i søgefeltet 2. Tryk på 'Search' knappen
Undtagelser	*Ingen*
Sluttilstand	Nye ture bliver hentet og vist i vinduet
Referencer	Se kildekode under 1.1.7 – SearchBtn_OnClick

Titel	Detaljeret visning af tur
Formål	At vise turen der blev klikket på i fuldskærm så der er plads til flere detaljer
Aktører	Bruger
Frekvens	Når aktøren klikker på en TripTile
Startbetingelser	Aktøren er logget ind og at hoved vinduet har indlæst ture fra databasen
Beskrivelse	1. Tryk på en TripTile
Undtagelser	Hvis der allerede er åbnet en tur i fuldskærm
Sluttilstand	Vinduet maksimeres og ruten bliver indlæst på et stort kort
Referencer	Se kildekode under 1.1.8 - TripItemsControl_OnMouseLeftButtonUp

Titel	Returner til simplificeret oversigt af ture
Formål	At gå tilbage til oversigt af ture og indlæse de samme TripTiles som blev vist sidste gang
Aktører	Bruger
Frekvens	Når aktøren klikker højre klik i fuldskærms visning
Startbetingelser	Aktøren er logget ind og har navigeret til fuldskærms visning
Beskrivelse	1. Højre klik i vinduet

Undtagelser	Hvis man allerede er i oversigten
Sluttilstand	Vinduet returnerer til oversigt af ture med indlæsning af TripTiles efter den seneste søgning fortaget
Referencer	Se kildekode under 1.1.9 - RouteViewerMap_OnMouseRightButtonUp

Raven-Android

Titel	Opret forbindelse til Arduino
Formål	Bluetooth forbindelse bliver skabt mellem Arduino og Android
Aktører	Bruger
Frekvens	Hver gang efter en bruger har lukket appen og vil fortage en logning
Startbetingelser	Ingen aktive Bluetooth forbindelser. Arduino med Bluetooth modul i telefonens rækkevidde.
Beskrivelse	1. Tryk på "open" i MainActivity
Undtagelser	En forbindelse er allerede skabt.
Sluttilstand	En forbindelse er skabt til Arduino
Referencer	Se kildekode under 6.1.3 – OpenBTBtn_OnClick

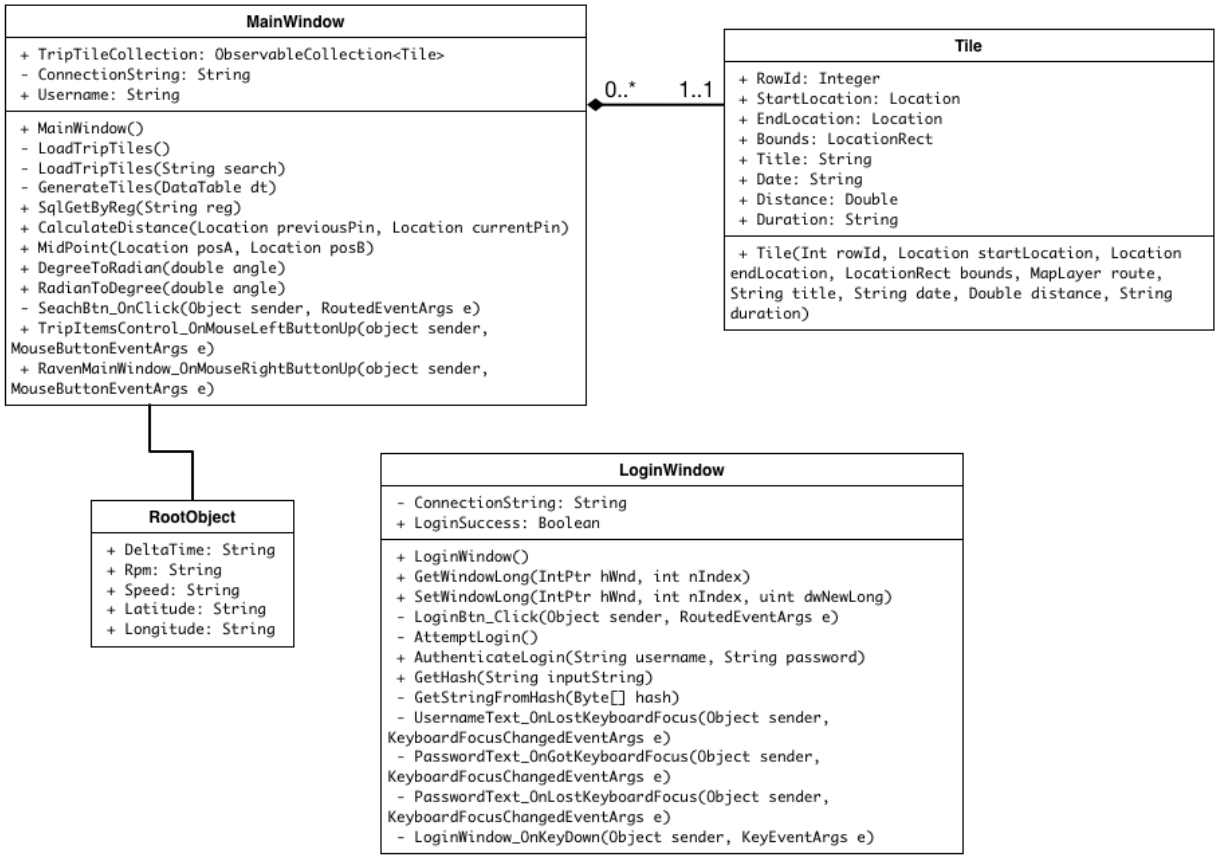
Titel	Start af logning
Formål	Start af indlæsning af Bluetooth data til jsonArray
Aktører	Bruger
Frekvens	Når brugeren aktivere logning.
Startbetingelser	En aktiv forbindelse til Bluetooth. Logning disabled
Beskrivelse	1. Tryk på knappen "Logging Disabled"
Undtagelser	Logning allerede aktiv. Ingen forbindelse til Bluetooth
Sluttilstand	Logning aktiv og en fast inkrementering af feltet "Lines in Array" hvert 2. sekund
Referencer	Se kildekode under 6.1.1.1 - onCheckedChanged.

Klasse Diagram

Et klasse diagram er en oversigt af en eller flere klasser. Et klasse diagram viser hvert klasses variabler, typer og metoder, det skrives også et tegn foran hvert element i diagrammet for at vise deres tilgængelighed, dvs. 'Private', 'Public', 'Protected' eller 'Internal'.

Et klasse diagram giver en god og hurtig forståelse af hvad der klassernes indhold er før man går dybt ned og analyserer koden. Klasse diagrammer kan være et godt værktøj til nye medarbejdere som starter på et allerede igangsat projekt.

Vi bruger standarden UML da det er den mest udbredte og kendte standard.



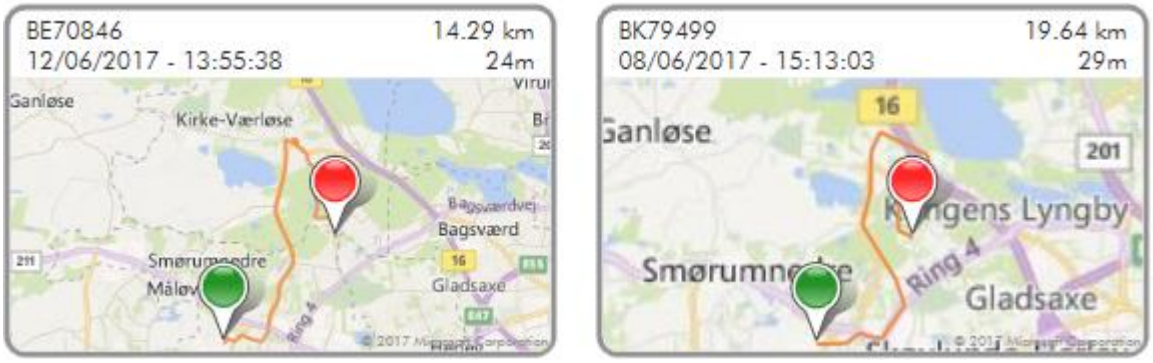
(fig. 7)

Kodebeskrivelse

Raven-Desktop

1.1.0 – MainWindow.xaml.cs

Er en klasse der indeholder fire globale variabler, TripTileCollection der indeholder ture af vores type 'Tile'(4.1.0).
Her er nogle eksempler på et 'TripTile'.



(fig. 8)

ConnectionString som indeholder addressen, database navnet og login til vores server, Username som vi bruger til første søgning af TripTiles og visning af hvem man er logget ind som i vindue titlen, Username bliver sat af LoginWindow. LastSearch som bliver brugt til at gemme den seneste søgning som brugeren har fortaget sig, grunden til at vi gemmer søgningen er fordi, når brugeren har klikket på et 'TripTile' viser det et stort kort omkring ruten, men når brugeren trykker højre klik så sender vi brugeren tilbage til hvor man kan se 'TripTiles', der bruger vi 'LastSearch' til at finde de samme 'TripTiles' som brugeren kiggede på sidst uden at brugeren behøver at søge igen.

1.1.1 – MainWindow

Er en constructor som laver et nyt LoginWindow(2.1.0) element, og så skjuler den sig selv så MainWindow hverken vises på skærmen eller i værktøjslinjen. Derefter hiver den LoginWindow frem og afventer at LoginWindow bliver lukket, afhængelig af 'LoginSuccess' boolean fra LoginWindow lukker den enten alle vinduer eller viser sig selv igen og opdaterer nogle variabler inden den begynder at indlæse TripTiles.

1.1.2 – LoadTripTiles

Er en metode som starter en forbindelse via en 'connection string' til vores Mysql server med udtaget "SELECT * FROM trips" hvilket betyder, hent alle linjer i 'trips' tabellen og læg det i et DataTable. Efter den har hentet informationerne kalder den GenerateTiles metoden og medsender DataTable'et.

1.1.3 – LoadTripTiles(string search)

Er en metode som næsten er helt ens med LoadTripTiles(1.1.2), denne metode laver bare udtrækket "SELECT * FROM trips WHERE CONCAT_WS('', time_started, time_ended, driver_username, driver_reg) LIKE '%{search}%'" hvis søgefeltet ikke er tomt. Derefter kalder den GenerateTiles(1.1.4) med DataTable'et.

1.1.4 – GenerateTiles

Er en metode som først tømmer 'TripTileSelection' og derefter begynder at løbe igennem alle rækker som den fik fra LoadTripTiles(1.1.2/1.1.3). For hver række fisker den informationer ud af rækken, såsom 'rowId', 'title', 'date', 'dateStart', 'dateEnd' og 'logs'.

Så tager den 'logs' og deserialiserer indholdet til en liste ved navn 'results' af 'RootObject' typen. RootObject er en klasse som benytter JSON attributter for at sortere data til det korrekte felt.

Efter den har deserialiseret loggen begynder den at tjekke alle resultaterne igennem for at finde fire koordinater. Det mest nordlige, sydlige, vestlige og østlige punkt af alle koordinaterne i 'results'

Efter den har fundet de fire koordinater laver den en 'LocationRect' ud fra de koordinater men med en forskydning for at synliggøre vores "knappenåle" som vi vil sætte på korterne. LocationRect kan bruges til at finde et midtpunkt og grænser for hvor meget kortene må zoomes ind.

Så danner den start og slut lokationer ud fra de første koordinater i listen og de sidste. De koordinater bruges til at lave vores grønne start knappenål og vores røde slut knappenål.

Herefter danner den et 'MapLayer' kaldet 'polyLineLayer' hvilket er et grafisk lag du kan smide hen over kortet, det bruger vi til at vise den kørte rute, men først skal det tegnes. Det gør den ved at løbe alle indekser igennem 'results' og lave en 'MapPolyLine' imellem hvert indeks ved hjælp af indeksernes koordinater og tilføjer det til 'polyLineLayer' som et barn, til sidst har vi et 'MapLayer' som indeholder vores rute.

Derefter formaterer den datoen til at se pænere ud.

Så laver den en 'distance' variabel og udregner den totale distance for ruten ved at løbe igennem 'results' og udregne distancen imellem to koordinater ved at kalde CalculateDistance(1.1.6) sammen med to 'Locations' og lægger den returnerede meter værdi til 'distance' for hvert loop. Efter den er færdig med for løkken dividerer den 'distance' med 1000 for at konvertere til km.

Derefter laver den en 'duration' streng og udregner turens varighed ved at finde differencen imellem 'dateStart' og 'dateEnd', og formaterer 'duration' til en bedre format for brugeren at læse.

Til sidst skaber den en ny Tile(4.1.0) i vores 'TripTileCollection' med de relevante informationer som den har forberedt.

Eksempel:

```
TripTileCollection.Add(new Tile(rowId, startLocation, endLocation, bounds, polyLineLayer, title, date, distance, duration));
```

1.1.5 – RouteViewerLoadTrip

Er en metode som modtager et 'tripId' som integer og laver et "SELECT * FROM trips WHERE id={tripId}" udtræk fra vores SQL server ved hjælp af 'tripId'.

Den tager loggen fra den række der matcher id'et og deserialiserer indholdet til en liste som den derefter retunerer til det sted 'RouteViewerLoadTrip' blev kaldt fra.

1.1.6 – CalculateDistance

Er en metode som modtager to 'Locations' (previousPin og currentPin), den tager deres højdegrad og længdegrad, og laver to 'GeoCoordinate'. 'GeoCoordinate' klassen har en metode som hedder 'GetDistanceTo' som udregner distancen for os, det retunerer distancen i meter til det sted 'CalculateDistance' blev kaldt fra.

1.1.7 – SearchBtn_OnClick

Er en metode som bliver kørt når brugeren klikker på vores 'Search' button i UI.

Den kalder LoadTripTile(1.1.3) hvis vores søgefelt ikke er tomt og medsender teksten i vores søgefelt. Hvis feltet er tomt så kalder den LoadTripTile(1.1.2).

Derefter gemmer den indholdet af søgefeltet i 'LastSearch'.

1.1.8 – TripItemsControl_OnMouseLeftButtonUp

Er en metode som skjuler 'TripTileGrid', synliggøre 'RouteViewerGrid' og sætter vinduet i fuldskærms tilstand.

Derefter får det fat i det TripTile element som der blev trykket på og caster det til en Tile(4.1.0). Ved at caste det til en 'Tile' kan vi tilgå datakonteksten og læse indholdet på det element der blev trykket på, den gemmes lokalt som 'item'.

Den tager 'RowId' fra elementet og benytter RouteViewerLoadTrip(1.1.5) til at slå den specifikke rute op og deserialisere loggen til en liste ved navn 'locations'.

Ved brug af 'locations' danner den to nye knappenåle, grøn start og rød slut.

Så danner den et 'MapLayer' ved navn 'polyLineLayer' og løber alle indekser igennem i en for løkke hvor den laver en 'MapPolyLine' imellem alle indekser ved brug af indeksernes lokationer og tilføjer det til 'polyLineLayer' som et barn.

Derefter laver den en 'LocationRect' ud fra 'item's værdier og tilføjer en forskydning så det ser bedre ud på et stort kort, den gemmes lokalt som 'bounds'.

Efter det rydder den vores 'TripTileCollection' for at vores 'RouteViewerMap' (som er det eneste kort som skal bruges/vises på det her tidspunkt) kører mere jævnt.

Så sætter den 'RouteViewerMap' kortets synspunkt ved brug af 'bounds', tilføjer 'polyLineLayer' som et barn på kortet og tilføjer start og slut knappenålene til kortet.

1.1.9 – RouteViewerMap_OnMouseRightButtonUp

Er en metode som synligøre 'TripTileGrid', skjuler 'RouteViewerMap' og formindsker vinduet.

Så rydder den 'RouteViewerMap' og kalder LoadTripTiles(1.1.3) og medsender 'LastSearch' fra MainWindow(1.1.0) så at brugeren ender der hvor de var sidst.

1.2.0 - RootObject

Er en klasse som indeholder de forskellige properties vi burger i vores JSON logs.

Hvert property har sin egen attribute som gør det let for os når vi deserialiserer vores logs. Attributterne bruges til at associere et property navn til en anden property.

Det vil sige at når vi deserialiserer vores logs, så ser den f.eks. navnet "DeltaTime", når den ser det navn ved den straks at den property's værdi hører til i "min" egen property (Som vi I dette tilfælde kalder det samme, men navnene behøver ikke være ens).

Det vi bruger 'RootObject' til er at midlertidigt gemme loggene så vi har et objekt som er ekstremt let at læse fra ved at referere til en liste af typen 'RootObject'.

2.1.0 – LoginWindow.xaml.cs

Er en klasse som indeholder to globale variabler, ConnectionString som indeholder addressen, database navnet og login til vores server, LoginSuccess som er en boolean til at tjekke om det lykkedes brugeren at logge ind. Grunden til 'LoginSucess' er fordi MainWindow(1.1.1) har et event som bliver kørt når LoginWindow lukkes, og der tjekker den tilstanden på 'LoginSucess' og forsætter derfra.

2.1.1 – LoginWindow

Er en constructor som initialiserer sig selv.

2.1.2 – LoginBtn_Click

Er en metode som bliver kørt når brugeren klikker på 'Login' knappen i LoginWindow(2.1.0).

Den kalder metoden AttemptLogin(2.1.3).

2.1.3 – AttemptLogin

Er en metode som kalder AuthenticateLogin(2.1.4).

Hvis AuthenticateLogin returnerer 'true' så skifter den 'LoginSuccess' til 'true' og ændrer MainWindow(1.1.0) 'Username' til enten "DebugUser" eller det indtastede brugernavn (i lovercase) og derefter lukker vinduet.

Hvis AuthenticateLogin returnerer 'false' så giver den fejlmeddelelsen "Username or password is incorrect".

2.1.4 – AuthenticateLogin

Er en metode som modtager to parameter, brugernavn og password.

Først kalder den GetHash(2.1.5) for at generere en SHA512 krypteringen af 'password'.

Derefter opretter den forbindelse til vores sql server og udtrækker alle rækker i 'logins' tabellen, derefter løber den hver række igennem og tjekker om 'username' og SHA512 'password' er ens med rækken.

Metoden returnerer 'true' hvis den finder ens 'username' og SHA512 'password' på samme række. Eller returnerer 'false' hvis intet match blev fundet.

Grunden til at vi tjekker efter SHA512 kryptering er for ikke at have brugeres adgangskode stående i ren tekst. Opbevaring af adgangskoder i ren tekst er usikkert og uansvarligt.

2.1.5 – GetHash

Er en metode som modtager en 'inputString' som er en adgangskode.

Den tager adgangskoden og spytter det i et array som bytes, derefter sættes SHA512 til at kryptere hver byte i array'et og så kalder den GetStringFromHash(2.1.6) og får en samlet string tilbage som den returnerer til der GetHash blev kaldet fra.

2.1.6 – GetStringFromHash

Er en metode som modtager et byte array 'hash'.

Metoden bygger en ny streng 'result' og løber igennem hele byte array'et hvor den konverterer et byte til en to cifret hexadecimal string med store bogstaver og til sidst tilføjer det til 'result'.

Derefter returnerer den 'result' strengen til GetHash(2.1.5).

2.1.7 – UsernameText_OnLostKeyboardFocus

Er en metode som tømmer indholdet af 'UsernameText' kontrollen hvis der er en tom karakter når man klikker væk fra tekst boksen.

2.1.8 – PasswordText_OnGotKeyboardFocus

Er en metode som tømmer indholdet af 'PasswordText' kontrollen og gør klar til indtastninger, men kun hvis indholdet af 'PasswordText' matcher vores "PPPPPPPP" dummy tekst.

2.1.9 – PasswordText_OnLostKeyboardFocus

Er en metode som indsætter dummy teksten "PPPPPPPP" i 'PasswordText' kontrollen hvis indholdet er en tom streng.

2.1.10 – LoginWIndow_OnKeyDown

Er en metode som lytter efter Enter.

Hvis knappen Højre Shift er holdt nede samtidigt med at der bliver trykket på Enter, så bruger den vores fiktive 'development' bruger og bypasser vores login tjek ved straks at sige 'LoginSuccess' fra LoginWindow(2.1.0) er 'true', derefter sætter den MainWindow(1.1.0) 'Username' til at være "DebugUser" og så lukker den LoginWindow vinduet.

Hvis der bare bliver trykket Enter kalder den AttemptLogin(2.1.3).

3.1.0 – TileMap.xaml.cs

Er en klasse som bruger til at skabe kortet til et 'TripTile'.

3.1.1 – TileMap

Er en constructor som initialiserer sig selv.

3.1.2 – TileMap_OnLoaded

Er en metode som bruger casting af sin egen datakontekst til et Tile(4.1.0) objekt.

Ved brug af castingen fra datakontekst kan metoden helt selv finde ud af hvem den er i datatemplaten på vores 'TripItemsControl' kontrol i MainWindow(1.1.0) og herfra tilgå dens værdier.

Den bruger adgangen til de værdier til at danne to nye knappenåle, en grøn start og en rød slut knappenål.

Så tilføjer den, den tegnede rute fra castingen som et nyt barn af kortet, sætter knappenålene på kortet og så sætter den kortets synspunkt ved brug af castingens 'Bounds' variabel.

4.1.0 – Tile.cs

Er en klasse med 9 globale variabler som vi bruger til at opbevare vores information omkring 'TripTiles' i. Variablerne er RowId som bruges til at holde et unikt id nummer til sql tabel rækken, StartLocation som holder lat og lng af hvor turen startede, EndLocation som holder lat og lng af hvor turen sluttede, Bounds hvilket bruges til at sætte grænserne på hvad kortet i en TripTile(3.1.0) må se, Route hvilket indeholder en tegnet rute fra GenerateTiles(1.1.6), Title hvilket indeholder registrerings nummeret på en bil, Date som indeholder start dato og tid på turen, Distance hvilket indeholder km distancen på en tur og Duration hvilket indeholder en turs varighed.

4.1.1 – Tile

Er en constructor som modtager 9 parameter og sætter sine globale variablers værdi i Tile(4.1.0) til det den har modtaget.

Raven-Android

5.1.0 – LoginActivity.java

LogonActivity bliver brugt til at autentikere brugere og angive username videre til brug i MainActivity

5.1.1 – onCreate

onCreate virker som en slags constructor, ofte brugt til at initialisere variabler eller knytte variabler til UI elementer

5.1.2 – LoginBtn_OnClick

Denne funktion bliver kaldt hver gang brugeren trykker på login knappen. Den laver en ny instans af RavenAPI_CheckLogin(5.2.0 – RavenAPI_CheckLogin) som laver et API kald til vores apache server

5.2.0 – RavenAPI_CheckLogin

CheckLogin er en AsyncTask klasse det vil sige at den kan udfører arbejde i baggrunden, det er en god ide at bruge for API kald fordi hvis ens server er langsom eller andre faktorer gør processen langsommere vil hele applikationen ikke gå i stå

Et API kald er stort set bare en http forbindelse, som i en browser.

Hvis man går på http//google.com med sin browser sender ens browser et request, google's servere vil så modtage dette request og sende et respons tilbage med HTML koden for hjemmesiden.

Men et API kald ville ikke være specielt brugbart hvis det kun var muligt at modtage, man skal også kunne sende data.

Heldigvis er dette en mulighed ved hjælp af URL parametre, i PHP kan man tilføje et nøgleord og en værdi at tilføje til sin URL dette vil se ud som følgende.

http://example.com/script.php?key=helloworld

I dette eksempel vil vi kunne i PHP trække værdien "helloworld" ud fra nøglen key.

5.2.1 – onPreExecute

Denne funktion bliver ikke brugt men det er en standard funktion i klassen AsyncTask og bliver tit brugt til at præsentere en loading bar for brugeren.

5.2.2 – doInBackground

doInBackground er der hvor vi gør det meste i vores AsyncTask. Den starter med at lave en instans af URL ud fra vores API_URL som er http://raven-gps.com/check_login.php derefter bliver brugernavn og password tilsluttet som URL parametre så vores URL vil til sidst blive http://raven-gps.com/check_login.php?user=brugernavn&pass=password

Se eventuelt under 7.1.0 - Check_login.php

For dette kald bruger vi protokollen "GET", get er en standard protokol for http forbindelser.

Vores server vil så udføre et lookup i vores SQL og returnere enten et 1 for succes eller 0 hvis brugernavn eller password er forkert

5.2.3 – onPostExecute

Denne funktion bliver kaldt efter en task er blevet fuldført, og meget som dens søster-funktion onPreExecute bliver den meget ofte brugt til at afslutte loading bars. Den bliver også brugt til at håndtere hvad der skal gøres med de modtaget data, det er præcist hvad vi bruger den til i vores API kald – til at tjekke om vores svar var en succes eller der er sket en fejl.

6.1.0 – MainActivity.java

MainActivity er vores primære activity i applikationen. En activity i Android sammenhæng kan nok bedst forklares som en bid af applikationen, hvis man har en app som har mange forskellige sider som f.eks. en log ind side, en butik, indkøbskurv og indstillinger. Vil alle disse side være hver sin egen activity.

Activities kan skifte til hinanden på kryds og tværs ved brug af intents.

Hvis man forstiller sig en applikation som en bog med alle activities som en side hver, vil intents fungere som læserens fingre som skifter mellem sider.

Intents har en brugbar funktion for at dele information på tværs af applikationen, dette er kendt som extras og giver udvikleren mulighed for at tilknytte en værdi til et keyword, dette kan derefter blive hentet fra modtager-activity.

6.1.1 – onCreate

I denne onCreate tilslutter vi vores UI elementer til lokale variabler for nem adgang.

Username bliver hentet fra LoginActivity ved hjælp af en extra fra intent (6.1.0)

For mere information se 5.1.1 – onCreate

6.1.1.1 – onCheckedChanged

Android udvikling er specielt på visse punkter er platformen meget nytænkende og innovativt, men på andre punkter er det meget gammeldags og primitivt. ToggleButtons er et af de primitive punkter.

En ToggleButton i Android kan ikke blive tilknyttet en event fra UI designeren, så den bedste måde at bruge dem på er ved at tilføje en onCheckedEventListener til knappen i onCreate (6.1.1) sammen med en switch-case.

Hvis Android tilføjede muligheden for at angive to seperate funktioner til Enabled og Disabled, ville verden være et bedre sted.

6.1.1.1.1 – onClick

Denne funktion er en del af en dialog boks der kommer frem efter brugeren har logget en køretur og afslutter loggen, onClick bliver kaldt hvis brugeren vælger "Upload to SQL". Et API kald bliver aktiveret (8.1.0) og den loggede køretur vil blive uploadet til SQL til brug i Raven-Desktop.

6.1.2 – CloseBTBtn_OnClick

Denne funktion er til modsætning for onCheckedChanged (6.1.1.1) blevet assigned gennem UI designeren som alle events burde være.

Denne knap lukker for alle åbne forbindelser til Bluetooth, hvis ingen forbindelser er aktive vil intet ske.

6.1.3 – OpenBTBtn_OnClick

Open knappen kalder to funktioner, findBT (6.1.4) og openBT (6.1.5) der først forsøger at finde vores Arduino hvis den kan findes prøver den at åbne en forbindelse.

6.1.4 – findBT

Prøve at finde Arduino ved at hente en liste over parrede enheder og tjekke om Arduino er i range.

6.1.5 – openBT

Bluetooth er en fascinerende teknologi, en teknologi til alle tænkelige kommunikations protokoller.

I vores openBT funktion bruger vi en UUID (Universally Unique Identifier) der definere hvilken kommunikations protokol Bluetooth skal bruge, i vores situation har vi valgt seriel protokollen.

Men vi kunne ændre den UUID og definere Android som et tastatur, mus eller noget helt tredje.

Herefter starter vi en sokkel forbindelse til Arduino og kalder beginListenForData (6.1.6) som læser enheden for data

6.1.6 - beginListenForData

Denne funktion starter de tråde der kører i baggrunden imens en aktiv Bluetooth forbindelse er i gang. Den bliver brugt til at 'bygge' et samhørigt svar fra Arduino ved at samle alle Bluetooth packets.

Når et fuldt svar er blevet opfanget, starter tråden en ny tråd for så at sende data videre til loadJson (6.1.9).

En ny tråd er nødvendig for ikke at sløve kommunikations tråden. Hvis dette ikke skete ville kommunikationen være hakkende og ustabil – konstant tab a packets

6.1.7 – WriteLog

WriteLog er et shortcut, det er nemmere at skrive WriteLog("hello") end Android.util.Log.e("info", "Hello").

For det meste kun brugt til debugging for at hurtigt kunne skrive en one-liner der fortæller hvad gik galt et sted.

6.1.8 – SetTextFromPID

SetTextFromPID tager tre parametre, et textview, en string for PID og et json objekt.

Disse bliver brugt til at udtrække en værdi fra JSON med nøglen i PID og sætte texten på textview til det.

Sagt på en anden måde, den sætte teksten på vores liste af værdier i MainWindow, denne liste kan også ses i vores brugervejledning under Forklaring af UI Elementer.

6.1.9 – loadJson

LoadJson bliver kaldt fra beginListenForData (6.1.6) og bliver brugt for at håndtere de data vi modtager fra Bluetooth.

Den starter med at initialisere et nyt JSON objekt ud fra dens input parameter, de vil sige de data vi modtager fra beginListenForData (6.1.6).

Derefter bliver SetTextFromPID (6.1.8) kørt for at forsøge at sætte de værdier der hører til diverse TextViews.

Hvis logning er aktivt, bliver de modtaget data gemt i vores jsonArray for senere brug.

6.1.10 – sendData

sendData bliver ikke brugt i vores færdige applikation, men den blev brugt i starten af udviklingsfasen for at debugge forskellige funktioner på Arduino.

Det er en simpel funktion der tager tekst fra et textview og skriver til output bufferen der blev genereret i openBT (6.1.5)

6.1.11 – closeBT

Denne funktion bliver brugt til at lukke bluetooth forbindelsen på en pæn måde.

Ved at lukke vores input- og outputbuffers og lukke forbindelses soklen.

6.2.0 – RavenAPI_LogTrip

LogTrip er en af vores API kald der benytter insert.php (8.1.0) og indsætter vores loggede data til SQL ved hjælp af et POST request.

Et POST request er ligesom et GET request bortset fra at POST har mulighed for at sende parametre på flere måder en kun over URL.

Det giver os mulighed for at sende større stykker data uden at skrive det bag URL.

Den måde vi sender vores data på er gennem et JSON objekt, det gør det nemt at behandle på PHP siden og er et meget "clean" format.

For mere information om hvordan et API fungere se forklaringen under 5.2.0

6.2.1 – onPreExecute

Som skrevet før bliver onPreExecute brugt til at vise brugeren at der bliver hentet data eller på andre måder arbejdet i baggrunden. I LogTrip er dette dog ikke nødvendigt da vores SQL statement tager ingen tid.

6.2.2 – doInBackground

Denne funktion er næsten identisk med doInBackground (5.2.2) i CheckLogin.

Forskellen mellem disse er den protokol vi benytter, hvor vi bruger POST i stedet for GET og vores metode på at sende information til serveren.

Vi opretter et json objekt og putter vores relevante data ind i dette objekt hvor det så vil blive skrevet til vores output buffer og sendt med i vores request.

6.2.3 – onPostExecute

I post execute er der ikke så meget der sker. Vores API kald er et insert så der er ikke noget data at behandle. Det vi har valgt at gøre er at sende svaret fra serveren til brugeren for at give feedback på om upload var succesfuldt.

Konklusion

Koden

Da vi først valgte vores projekt regnede vi med at vi kun havde brug for emnerne SQL, Arduino, Android og WPF, pludseligt stod vi og inddragede flere emner såsom PHP, C++ og mere.

Vi løb ind i mange problemer, såsom hvordan vi skulle lagre data, kommunikere mellem de forskellige platforme og GPS funktionalitet.

Vi har lært en masse om hvordan man kan løse de problemer men også bedre alternativer på løsninger vi allerede kendte.

Rapporten

Vi har lært hvor vigtigt det er at starte på rapporten tidligt i projektet, ikke nødvendigvis fordi at man ikke kan nå det hvis man først begynder midt i projektet men fordi at det giver et bedre overskud og mindre stress over at skrive rapport. Ved at starte tidligt kan man skiftevis skrive rapport eller kode, så kommer der variation i sit arbejde og så har man ikke en kæmpe opgave men til gengæld en delvist færdig rapport at færdigøre til sidst.

Vi er meget glade for at kunne have en "forsmag" på svendeprøven og vi ved nu hvordan vi bedre kan arbejde til næste gang.

Kildekode

Desktop

1.1.0 - MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data;
using System.Windows;
using System.Windows.Controls;
using System.Device.Location;
using System.Globalization;
using System.Windows.Input;
using System.Windows.Media;
using MySql.Data.MySqlClient;
using Microsoft.Maps.MapControl.WPF;
using Newtonsoft.Json;
using Raven.Windows;

namespace Raven {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow {
        public static ObservableCollection<Tile> TripTileCollection { get; set; } =
            new ObservableCollection<Tile>(); // Collection of TripTiles, using custom Tile type

        private const string ConnectionString = "Server=raven-
gps.com;Database=raven;Uid=root;Pwd=#### ";
        public static string Username = "";
        public static string LastSearch = "";
```

1.1.1 – MainWindow

```
        public MainWindow() {
            // Initiate LoginWindow element
            var loginWindow = new LoginWindow();
            Hide();
            loginWindow.Show();
            InitializeComponent();

            loginWindow.Closed += delegate {
                if (loginWindow.LoginSuccess) {
                    Title = $"Raven - Logged in as {Username}";
                    Show();
                    if (Username == "DebugUser") {
                        LoadTripTiles();
                    }
                    else {
                        LoadTripTiles(Username);
                        LastSearch = Username;
                    }
                }
                else {
                    Close();
                }
            };
        }
```

1.1.2 – LoadTripTiles

```
private void LoadTripTiles() {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();
        command.CommandText = "SELECT * FROM trips";

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            GenerateTiles(dt);
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }
}
```

1.1.3 – LoadTripTiles (med søge parameter)

```
private void LoadTripTiles(string search) {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();

        if (search == String.Empty) {
            command.CommandText = "SELECT * FROM trips";
        }
        else {
            command.CommandText = $"SELECT * FROM trips WHERE CONCAT_WS('', time_started,
time_ended, driver_username, driver_reg) LIKE '%{search}%'";
        }

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            GenerateTiles(dt);
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }
}
```

1.1.4 – GenerateTiles

```
public void GenerateTiles(DataTable dt) {
    TripTileCollection.Clear();
    foreach (DataRow row in dt.Rows) {
        try {
            var rowId = int.Parse(row["id"].ToString());
            var logs = row["log_file"].ToString();
            var title = row["driver_reg"].ToString();
            var date = row["time_started"].ToString();
            var dateStart = DateTime.Parse(row["time_started"].ToString());
            var dateEnd = DateTime.Parse(row["time_ended"].ToString());
```

```
            var results = JsonConvert.DeserializeObject<List<RootObject>>(logs);

            // Set Bounds to fit all pins
            var mostNorth = double.Parse(results[0].Latitude, CultureInfo.InvariantCulture);
            var mostSouth = double.Parse(results[0].Latitude, CultureInfo.InvariantCulture);
            var mostEast = double.Parse(results[0].Longitude, CultureInfo.InvariantCulture);
            var mostWest = double.Parse(results[0].Longitude, CultureInfo.InvariantCulture);

            for (var i = 1; i < results.Count; i++) {
                var lat = double.Parse(results[i].Latitude, CultureInfo.InvariantCulture);
                var lng = double.Parse(results[i].Longitude, CultureInfo.InvariantCulture);

                if (lat > mostNorth) {
                    mostNorth = lat;
                }
                else if (lat < mostSouth) {
                    mostSouth = lat;
                }
                if (lng > mostEast) {
                    mostEast = lng;
                }
                else if (lat < mostWest) {
                    mostWest = lng;
                }
            }
            var bounds = new LocationRect(new Location(mostNorth + 0.015, mostWest + 0.0075),
new Location(mostSouth - 0.002, mostEast - 0.0075));

            // Set start and end locations
            var startLocation = new Location(double.Parse(results[0].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[0].Longitude, CultureInfo.InvariantCulture));
            var endLocation = new Location(double.Parse(results[results.Count - 1].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[results.Count - 1].Longitude,
CultureInfo.InvariantCulture));

            // Create MapLayer of driven route
            var polyLineLayer = new MapLayer(); // Layer used only for MapPolyLines for easier cleaning
            for (var i = 1; i < results.Count; i++) {
                var polyLine = new MapPolyline();
                var colourBrush = new SolidColorBrush {Color = Color.FromRgb(232, 123, 45)};
                polyLine.Stroke = colourBrush;
                polyLine.StrokeThickness = 2;
                polyLine.Opacity = 1.0;

                polyLine.Locations = new LocationCollection {
                    new Location(double.Parse(results[i - 1].Latitude, CultureInfo.InvariantCulture),
double.Parse(results[i - 1].Longitude, CultureInfo.InvariantCulture)),
                    new Location(double.Parse(results[i].Latitude, CultureInfo.InvariantCulture),
double.Parse(results[i].Longitude, CultureInfo.InvariantCulture))
                };
                polyLineLayer.Children.Add(polyLine); // Adds a new line to the layer
            }

            // Format date
            date = date.Replace('-', '/');
            date = date.Replace(" ", " - ");

            // Calculate distance
            var distance = 0.0;
            for (var i = 1; i < results.Count; i++) {
```

```
                var oldIndex = new Location(double.Parse(results[i - 1].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[i - 1].Longitude, CultureInfo.InvariantCulture));
                var newIndex = new Location(double.Parse(results[i].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[i].Longitude, CultureInfo.InvariantCulture));

                distance = distance + CalculateDistance(oldIndex, newIndex);
            }
            distance = distance / 1000; // Convert from meters to kilometers

            // Calculate duration
            // MAYBE Change to get difference between first and last index of 'results' collection
            string duration;
            var difference = dateEnd - dateStart;
            if (difference.TotalDays >= 1) {
                duration = difference.Days + "d " + difference.Hours + "h " + difference.Minutes + "m";
            }
            else if (difference.TotalHours >= 1) {
                duration = difference.Hours + "h " + difference.Minutes + "m";
            }
            else {
                duration = difference.Minutes + "m";
            }

            // Create TripTile
            TripTileCollection.Add(new Tile(rowId, startLocation, endLocation, bounds, polyLineLayer,
title, date, distance, duration));
        }
        catch (Exception) {
            //ignore
        }
    }
}
```

1.1.5 – RouteViewerLoadTrip

```
private List<RootObject> RouteViewerLoadTrip(int tripId) {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();
        command.CommandText = $"SELECT * FROM trips WHERE id={tripId}";

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            foreach (DataRow row in dt.Rows) {
                var logs = row["log_file"].ToString();
                return JsonConvert.DeserializeObject<List<RootObject>>(logs);
            }
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }

    return null;
}
```

1.1.6 – CalculateDistance

```
// Returns the distance (in metric meters) between two locations
```

```
        public static double CalculateDistance(Location previousPin, Location currentPin) {
            try {
                var firstCoordinate = new GeoCoordinate(previousPin.Longitude, previousPin.Latitude);
                var secondCoordinate = new GeoCoordinate(currentPin.Longitude, currentPin.Latitude);

                return firstCoordinate.GetDistanceTo(secondCoordinate);
            }
            catch (Exception) {
                return 0;
            }
        }
```

1.1.7 – SearchBtn_OnClick

```
        private void SearchBtn_OnClick(object sender, RoutedEventArgs e) {
            if (SearchDetailsBox.Text != string.Empty) {
                LoadTripTiles(SearchDetailsBox.Text);
            }
            else {
                LoadTripTiles();
            }

            LastSearch = SearchDetailsBox.Text;
        }
```

1.1.8 - TripItemsControl_OnMouseLeftButtonUp

```
        // TODO Generate pins with custom orange ellipse style
        // TODO Click event for pins that display all details of that pin on the right hand side
        private void TripItemsControl_OnMouseLeftButtonUp(object sender, MouseButtonEventArgs e) {
            // Visibility control
            TripTileGrid.Visibility = Visibility.Collapsed;
            RouteViewerGrid.Visibility = Visibility.Visible;
            RavenMainWindow.WindowState = WindowState.Maximized;

            // Load info
            var clickedItem = (FrameworkElement) e.OriginalSource;
            var item = (Tile) clickedItem.DataContext;
            var locations = RouteViewerLoadTrip(item.RowId);

            // Create Start/Stop pins
            var startLocation = new Pushpin {Background = Brushes.Green, Location = item.StartLocation,
Cursor = Cursors.Hand};
            var endLocation = new Pushpin {Background = Brushes.Red, Location = item.EndLocation,
Cursor = Cursors.Hand};

            // Create MapLayer of 'locations'
            var polyLineLayer = new MapLayer(); // Layer used only for MapPolyLines for easier cleaning
            for (var i = 1; i < locations.Count; i++) {
                var polyLine = new MapPolyline();
                var colourBrush = new SolidColorBrush {Color = Color.FromRgb(232, 123, 45)};
                polyLine.Stroke = colourBrush;
                polyLine.StrokeThickness = 3;
                polyLine.Opacity = 1.0;

                polyLine.Locations = new LocationCollection {
                    new Location(double.Parse(locations[i - 1].Latitude, CultureInfo.InvariantCulture),
double.Parse(locations[i - 1].Longitude, CultureInfo.InvariantCulture)),
                    new Location(double.Parse(locations[i].Latitude, CultureInfo.InvariantCulture),
double.Parse(locations[i].Longitude, CultureInfo.InvariantCulture))
                };
                polyLineLayer.Children.Add(polyLine); // Adds a new line to the layer
```

```
            }

            // Create new bounds
            var bounds = new LocationRect(new Location(item.Bounds.Center.Latitude - 0.0065,
item.Bounds.Center.Longitude), item.Bounds.Width + 0.05, item.Bounds.Height + 0.05);

            // Clear TripTileCollection for smoother RouteViewerMap
            TripTileCollection.Clear();

            // Setup Map
            RouteViewerMap.SetView(bounds);
            RouteViewerMap.Children.Add(polyLineLayer);

            RouteViewerMap.Children.Add(startLocation);
            RouteViewerMap.Children.Add(endLocation);
        }
```

1.1.9 - RouteViewerMap_OnMouseRightButtonUp

```
        private void RouteViewerMap_OnMouseRightButtonUp(object sender, MouseButtonEventArgs e) {
            // Visibility control
            TripTileGrid.Visibility = Visibility.Visible;
            RouteViewerGrid.Visibility = Visibility.Collapsed;
            RavenMainWindow.WindowState = WindowState.Normal;

            // Cleanup Map
            RouteViewerMap.Children.Clear();

            // Reload TripTiles from previous search
            LoadTripTiles(LastSearch);
        }
```

1.2.0 - RootObject

```
        public class RootObject {
            [JsonProperty(PropertyName = "DeltaTime")]
            public string DeltaTime { get; set; }

            [JsonProperty(PropertyName = "Rpm")]
            public string Rpm { get; set; }

            [JsonProperty(PropertyName = "Speed")]
            public string Speed { get; set; }

            [JsonProperty(PropertyName = "Lat")]
            public string Latitude { get; set; }

            [JsonProperty(PropertyName = "Lng")]
            public string Longitude { get; set; }
        }

    }
}
```

2.1.0 - LoginWindow.xaml.cs

```
using System;
using System.Data;
using System.Linq;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Text;
using System.Windows;
using System.Windows.Input;
using MySql.Data.MySqlClient;

namespace Raven.Windows {
    /// <summary>
    /// Interaction logic for LoginWindow.xaml
    /// </summary>
    public partial class LoginWindow {
        private const string ConnectionString = "Server=raven-
gps.com;Database=raven;Uid=root;Pwd=####";
        public bool LoginSuccess;
```

2.1.1 - LoginWindow

```
        public LoginWindow() {
            InitializeComponent();
        }
```

2.1.2 – LoginBtn_Click

```
        private void LoginBtn_Click(object sender, RoutedEventArgs e) {
            AttemptLogin();
        }
```

2.1.3 - AttemptLogin

```
        private void AttemptLogin() {
            // If login is corrrect changes LoginSuccess to true and closes LoginWindow
            if (AuthenticateLogin(UsernameText.Text.ToLower(), PasswordText.Password)) {
                LoginSuccess = true;
                MainWindow.Username = UsernameText.Text.ToLower();
                Close();
            }
            // If username or password is incorrect show error
            else {
                MessageBox.Show("Username or password is incorrect");
            }
        }
```

2.1.4 - AuthenticateLogin

```
        // Returns true or false depending on a SQL check for a login that matches username and password
        public bool AuthenticateLogin(string username, string password) {
            // Stores hash value of password in hash
            var hash = GetHash(password);

            var connection = new MySqlConnection(ConnectionString);
            var dt = new DataTable();
            connection.Open();

            try {
                var command = connection.CreateCommand();
                command.CommandText = "SELECT * FROM logins";
```

```
            using (var dr = command.ExecuteReader()) {
                dt.Load(dr);

                // Returns true using LINQ expression
                if (dt.Rows.Cast<DataRow>().Where(variable => variable.Field<string>("username")
== username).Any(variable => variable.Field<string>("password") == hash)) {
                    return true;
                }
            }
        }
        catch (MySqlException exception) {
            MessageBox.Show(exception.ToString());
            return false;
        }
        return false;
    }
```

2.1.5 - GetHash

```
    // Returns plain password converted to SHA512 encrypted as string
    public static string GetHash(string inputString) {
        var sha512 = SHA512.Create();
        var bytes = Encoding.UTF8.GetBytes(inputString);
        var hash = sha512.ComputeHash(bytes);
        return GetStringFromHash(hash);
    }
```

2.1.6 - GetStringFromHash

```
    // Returns a hash string built from a byte array
    private static string GetStringFromHash(byte[] hash) {
        var result = new StringBuilder();
        foreach (var t in hash) {
            result.Append(t.ToString("X2"));
        }
        return result.ToString();
    }
```

2.1.7 - UsernameText_OnLostKeyboardFocus

```
    // Empties UsernameText if there are no characters when losing focus
    private void UsernameText_OnLostKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
        if (string.IsNullOrEmpty(UsernameText.Text)) {
            UsernameText.Text = null;
        }
    }
```

2.1.8 - PasswordText_OnGotKeyboardFocus

```
    // Empties PasswordText if it contains dummy characters when getting focus
    private void PasswordText_OnGotKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
        if (PasswordText.Password == "PPPPPPPP") {
            PasswordText.Password = null;
        }
    }
```

2.1.9 - PasswordText_OnLostKeyboardFocus

```
    // Fills PasswordText with dummy characters when losing focus
    private void PasswordText_LostGotKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
```

```
        if (PasswordText.Password == "") {
            PasswordText.Password = "PPPPPPPP";
        }
    }
```

2.1.10 - LoginWindow_OnKeyDown

```
    // Key event that registers Enter or RShift + Enter
    private void LoginWindow_OnKeyDown(object sender, KeyEventArgs e) {
        switch (e.Key) {
            case Key.Enter:
                if (Keyboard.IsKeyDown(Key.RightShift)) {
                    LoginSuccess = true;
                    MainWindow.Username = "DebugUser";
                    Close();
                }
                else {
                    AttemptLogin();
                }
                break;
        }
    }
  }
}
```

3.1.0 - TileMap.xaml.cs

```
using System.Windows;
using System.Windows.Media;
using Microsoft.Maps.MapControl.WPF;

namespace Raven.Controls {
    /// <summary>
    /// Interaction logic for TileMap.xaml
    /// </summary>
    public partial class TileMap {
```

3.1.1 - TileMap

```
        public TileMap() {
            InitializeComponent();
        }
```

3.1.2 – TileMap_OnLoaded

```
        private void TileMap_OnLoaded(object sender, RoutedEventArgs e) {
            var startLocation = new Pushpin { Background = Brushes.Green, Location =
((Tile)DataContext).StartLocation };
            var endLocation = new Pushpin { Background = Brushes.Red, Location =
((Tile)DataContext).EndLocation };

            Children.Add(((Tile)DataContext).Route);
            Children.Add(startLocation);
            Children.Add(endLocation);
            SetView(((Tile)DataContext).Bounds);
        }
    }
}
```

4.1.0 – Tile.cs

```
using Microsoft.Maps.MapControl.WPF;
```

```
namespace Raven {
    public class Tile {
        public int RowId { get; set; }
        public Location StartLocation { get; set; }
        public Location EndLocation { get; set; }
        public LocationRect Bounds { get; set; }
        public MapLayer Route { get; set; }
        public string Title { get; set; }
        public string Date { get; set; }
        public double Distance { get; set; }
        public string Duration { get; set; }
```

4.1.1 - Tile

```
        public Tile(int rowId, Location startLocation, Location endLocation, LocationRect
bounds, MapLayer route, string title, string date, double distance, string duration) {
            RowId = rowId;
            StartLocation = startLocation;
            EndLocation = endLocation;
            Bounds = bounds;
            Route = route;
            Title = title;
            Date = date;
            Distance = distance;
            Duration = duration;
        }
    }
}
```

Android

5.1.0 - LoginActivity.java

```
package com.raven_gps.raven;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class LoginActivity extends AppCompatActivity {

    // definition of UI elements to be assigned later
    EditText usernameText;
    EditText passwordText;
```

5.1.1 - onCreate

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

        // assign variables with actual ids in interface
        usernameText = (EditText) findViewById(R.id.UsernameText);
```

```
        passwordText = (EditText) findViewById(R.id.PasswordText);


    }
```

5.1.2 – LoginBtn_OnClick

```
    public void LoginBtn_OnClick(View v) {
        // Authentication of user by using an API Call
        new RavenAPI_CheckLogin(
                usernameText.getText().toString(),
                passwordText.getText().toString());
    }
```

5.2.0 – RavenAPI_CheckLogin

```
    // RavenAPI Authentication of Login Details
    private class RavenAPI_CheckLogin extends AsyncTask<Void, Void, String> {

        // API Script URL
        private String API_URL = "http://raven-gps.com/check_login.php";
        // Initialization of variables for use in constructor
        private String username;
        private String password;

        // Constructor of CheckLogin, assigns given variables to local variables
        RavenAPI_CheckLogin(String username, String password){
            this.username = username;
            this.password = password;
        }
```

5.2.1 - onPreExecute

```
        protected void onPreExecute() {
            /*
         * this is not used but is normally used to invoke a loadingbar
         * or other form of load indicator for the user
         */
        }
```

5.2.2 - doInBackground

```
        protected String doInBackground(Void... urls) {
            try {
                // forms the URL needed for executing the API call
                URL url = new URL(API_URL +
                        "?user=" + username +
                        "&pass=" + password);

                // Logs Which user is logged in, this is only accessible from a debug shell
                Log.e("AUTH", "USERNAME: " + username + " PASS: " + password);
                // Establishes an http connection to the server through the url defined earlier
                HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
                // Tells server this request is of type GET
                urlConnection.setRequestMethod("GET");

                // This is the meat and potatoes of the API call, this is reading what is being spit out of the server
                try {
                    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new StringBuilder();
                    String line;
                    // This is reading each line, basicly going "If there is anything to read I'll write it to the buffer followed by
a newline (\n)"
                    while ((line = bufferedReader.readLine()) != null) {
                        stringBuilder.append(line).append("\n");
```

```
                }
                // When there is no more to read, close the connection neatly
                bufferedReader.close();
                return stringBuilder.toString();
            }
            // If everything crashes and burns close connection
            finally{
                urlConnection.disconnect();
            }
        }
        // if anything goes wrong tell the developer
        catch(Exception e) {
            Log.e("ERROR", e.getMessage(), e);
            return null;
        }
    }
```

5.2.3 - onPostExecute

```
    // On post execute is like pre execute, only later
    // This is where we handle the data after we have talked to the server
    protected void onPostExecute(String response) {
        // Response is what the server told us which is a json response
        // formatted like the following
        // {"success": 1} or {"success": 0}

        try{
            // Log the response to the developer
            Log.e("INFO", response);

            // if the response has a one in it it's highly likely a positive response.
            // We have tried to make it check the actual json value. but this works fine.
            boolean resBool = response.contains("1");

            // if the server responded positively
            if (resBool){
                // prepare a new intent for switching to MainActivity
                // Normally intents is written like: new Intent(this, [otherActivity.class])
                // But because we are still in the API call we have to cast to our mother class
                Intent i = new Intent(LoginActivity.this, MainActivity.class);
                // tell the intent which user is authenticated
                i.putExtra("username", username);
                startActivity(i);
            }else{
                // Tell the devs what resbool is, this is still here since we used it for debugging purposes
                Log.e("TEST", String.valueOf(resBool));
                // Write a message on screen telling the user something went wrong
                Toast.makeText(LoginActivity.this, "Login Failed", Toast.LENGTH_SHORT).show();
            }
        }catch(Exception e){
            // If everything fails, log an error message to the developer
            Log.e("POSTERROR",e.getMessage());
            // give the same message to the user, users arent dumb
            Toast.makeText(LoginActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
  }
}
```

6.1.0 - MainActivity.java

```
package com.raven_gps.raven;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.CompoundButton;
import android.widget.GridLayout;
import android.widget.TextView;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ToggleButton;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Set;
import java.util.UUID;

public class MainActivity extends Activity {

    //Definition of variables to be later assigned during onCreate()
    TextView date;
    TextView time;
    TextView kmh;
    TextView rpm;
    TextView lat;
    TextView lng;
    TextView arrayCount;
    ToggleButton loggingToggleBtn;
    TextView log;
    GridLayout gridLayout;
    TextView myLabel;
    EditText myTextbox;
    BluetoothAdapter mBluetoothAdapter;
    BluetoothSocket mmSocket;
```

```
BluetoothDevice mmDevice;
OutputStream mmOutputStream;
InputStream mmInputStream;
Thread workerThread;
byte[] readBuffer;
TextView Log;
int readBufferPosition;
int counter;

// these are used for storing the Intent.extras bundle and username string from LoginActivity
String username;
Bundle extras;

// This is the logging array that is used to store the json data from raven dev kit
JSONArray jsonArray;

// This is used to indicate if the bluetooth connection should be stopped
boolean stopWorker;
```

6.1.1 - onCreate

```
// the activity' on create function, this is a standard function in all android activities
// it works like a constructor normally used for assigning variables to UI elements.
// This is not different in our application
@Override
public void onCreate(Bundle savedInstanceState) {
    // Standard code needed by onCreate()
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // assign username gotten from loginactivity to a local variable
    extras = getIntent().getExtras();
    username = "rwejlgaard";
    // tell developer which user is logged in
    WriteLog("AUTHUSER: " + username);

    // assign UI elements to pre defined variables
    gridLayout = (GridLayout) findViewById(R.id.gridLayout);
    date = (TextView) findViewById(R.id.date);
    time = (TextView) findViewById(R.id.time);
    kmh = (TextView) findViewById(R.id.kmh);
    rpm = (TextView) findViewById(R.id.rpm);
    lat = (TextView) findViewById(R.id.lat);
    lng = (TextView) findViewById(R.id.lng);
    loggingToggleBtn = (ToggleButton) findViewById(R.id.LoggingToggleBtn);
    arrayCount = (TextView) findViewById(R.id.arrayCount);
    log = (TextView) findViewById(R.id.log);

    // Fun Fact: toggleButtons are not allowed to be assigned an event through the UI designer
    // So in order to assign an event to a togglebutton it has to be done in a horribly old and
unaesthetic way
    loggingToggleBtn.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
```

6.1.1.1 - onCheckedChanged

```
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
```

```
                    switch (String.valueOf(isChecked).toUpperCase()) {
                        case "TRUE":        // ENABLED
                            //if logging is enabled make sure to disengage the system lockscreen
                            // to not accidentally close the bluetooth connection
                            // Some phones can close the bluetooth connection by locking the
screen

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

                            // reinitialize jsonArray to clear any previously saved data
                            jsonArray = new JSONArray();
                            break;
                        case "FALSE":       // DISABLED
                            // Reengage the system lockscreen to not drain the device battery

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
                            //Start building the dialogbox
                            AlertDialog.Builder adb = new
AlertDialog.Builder(MainActivity.this);
                            adb.setTitle("Save File");

                            // Bind a new button with the label "Upload to SQL" to an onclick
listener
```

6.1.1.1.1 - onClick

```
                            adb.setPositiveButton("Upload To SQL", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                    // dump the logged data to the log in the app
                                    log.setText(jsonArray.toString());
                                    try {
                                        // Call RavenAPI to insert the data
                                        new RavenAPI_LogTrip(
                                                "06/8/2017-15:13:03",
                                                "06/8/2017-15:42:18",
                                                username,
                                                "BK79499",
                                                jsonArray.toString()
                                        ).execute();
                                    }catch (Exception e){
                                        // if something fails tell the dev
                                        WriteLog("Upload To SQL failed");
                                    }
                                }
                            });

                            //set a cancel button to discard the saved data
                            adb.setNegativeButton("Discard", null);
                            adb.setCancelable(false);
                            adb.show();
                            break;
                    }
                }
            });
```

```
    }
```

6.1.2 – CloseBTBtn_OnClick

```
    public void CloseBTBtn_OnClick(View v) {
        try {
            closeBT();
        } catch (IOException ex) {
        }
    }
```

6.1.3 – OpenBTBtn_OnClick

```
    public void OpenBTBtn_OnClick(View v) {
        try {
            //first find the device
            findBT();
            // then establish connection
            openBT();
        } catch (Exception ex) {
            //if something fails show the user
            Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
```

6.1.4 – findBT

```
    // This function throws an exception if it fails, it makes the function try catch-able
    void findBT() throws Exception {

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        // if the current device has no bluetooth radio
        if (mBluetoothAdapter == null) {
            WriteLog("No bluetooth adapter available");
        }

        // if bluetooth is disabled, offer to enable it
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, 0);
        }

        // get a list of paired devices
        Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
        // if there is more that 0 devices
        if (pairedDevices.size() > 0) {
            //foreach device in list
            for (BluetoothDevice device : pairedDevices) {
                // HC-06 is the name of the bluetooth module in raven dev kit
                if (device.getName().equals("HC-06")) {
                    mmDevice = device;
                    WriteLog("Bluetooth Device Found");
                    break;
                }
            }
        }
```

```
    }
```

6.1.5 - openBT

```
    void openBT() throws Exception {


        //DONT CHANGE, THIS DEFINES THE BLUETOOTH SERIAL PROTOCOL
        // Bluetooth is kind of strange you have to tell it which protocol to use
        // this is the protocol code of serial communication
        // But it's also possible to tell it you are a keyboard, mouse, apache helicopter you can be
anything in software!
        UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

        // Send the protocol UID to BT and establish a socket connection
        mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
        WriteLog("Initializing Socket... Please wait");
        mmSocket.connect();

        // Connection established now we get the I/O buffers
        WriteLog("Socket connected!");
        mmOutputStream = mmSocket.getOutputStream();
        mmInputStream = mmSocket.getInputStream();

        WriteLog("Listening..");

        //Listen for data
        beginListenForData();

        WriteLog("Bluetooth Opened");
    }
```

6.1.6 - beginListenForData

```
    void beginListenForData() {

        //setup of useful variables for later use
        final Handler handler = new Handler();
        final byte delimiter = 10; //This is the ASCII code for a newline character
        stopWorker = false;
        readBufferPosition = 0;
        readBuffer = new byte[1024];

        // initializes a Thread for reading info incomming from BT
        workerThread = new Thread(new Runnable() {
            public void run() {
                //checks if it's allowed to run
                while (!Thread.currentThread().isInterrupted() && !stopWorker) {
                    try {
                        // get any data that we can pull
                        int bytesAvailable = mmInputStream.available();
                        // checks if there actually is any data
                        if (bytesAvailable > 0) {
                            byte[] packetBytes = new byte[bytesAvailable];
                            // read available data
                            mmInputStream.read(packetBytes);

                            // loops through everything that was recieved
```

```
                                        for (int i = 0; i < bytesAvailable; i++) {
                                            byte b = packetBytes[i];
                                            // if the current data is a newline
                                            if (b == delimiter) {
                                                byte[] encodedBytes = new
byte[readBufferPosition];
                                                System.arraycopy(readBuffer, 0, encodedBytes,
0, encodedBytes.length);
                                                // form a full string of all the recieved data
                                                final String data = new String(encodedBytes,
"US-ASCII");
                                                readBufferPosition = 0;
                                                // post the full recieved string to a thread to handle
it
                                                handler.post(new Runnable() {
                                                    public void run() {
                                                        // parses the recieved string
                                                        loadJson(data);
                                                    }
                                                });
                                            } else {
                                                readBuffer[readBufferPosition++] = b;
                                            }
                                        }
                                    }
                                } catch (IOException ex) {
                                    // if anything fails stop the connection
                                    stopWorker = true;
                                }
                            }
                        }
                    });

                    // start the thread
                    workerThread.start();
                }
```

6.1.7 - WriteLog

```
    public void WriteLog(String str) {
        // most logging functions in pretty much every language uses tags as a way
        // to filter some log entries from others
        // this log entry for an example sets the tag as info
        android.util.Log.println(android.util.Log.VERBOSE, "info", str);
    }
```

6.1.8 - SetTextFromPID

```
    //this tries to get the value from json with a given key
    // and sets it as the text of the TextView v
    void SetTextFromPID(TextView v, String PID, JSONObject j) {
        // is the key even in the json?
        if (j.has(PID)) {
            try {
                //Sets the text of the textview
```

```
                v.setText(String.valueOf(j.get(PID)));
            } catch (JSONException e) {
                // if the key is not in json, tell the log
                WriteLog(e.toString());
            }
        } else {
            // if everything fails, set text to null
            v.setText("Null");
        }
    }
```

6.1.9 - loadJson

```
    // this is the function that handles the string we get from each packet from bluetooth
    void loadJson(String json) {
        try {
            JSONObject j = new JSONObject(json);

            // set a textview text to a specified key in the json object
            SetTextFromPID(time, "DeltaTime", j);
            SetTextFromPID(rpm, "Rpm", j);
            SetTextFromPID(kmh, "Speed", j);
            SetTextFromPID(lat, "Lat", j);
            SetTextFromPID(lng, "Lng", j);
            SetTextFromPID(date, "Date", j);
            SetTextFromPID(time, "Time", j);

            // if logging is enabled save the json
            if (loggingToggleBtn.isChecked()) {
                jsonArray.put(j);
                // update the array count
                arrayCount.setText(String.valueOf(jsonArray.length() - 1));
            }

        } catch (JSONException e) {
            // if the json format is incorrect tell the log
            android.util.Log.println(android.util.Log.VERBOSE, "error", e.toString());
        }
    }
```

6.1.10 - sendData

```
    // unused sendData function, it actually works. but is not used
    // This could in theory be used to set a custom name of the bluetooth module
    void sendData() throws IOException {
        String msg = myTextbox.getText().toString();
        msg += "\n";
        mmOutputStream.write(msg.getBytes());
        WriteLog("Data Sent");
    }
```

6.1.11 - closeBT

```
// closes the active bluetooth connection neatly
void closeBT() throws IOException {
        stopWorker = true;
        mmOutputStream.close();
        mmInputStream.close();
        mmSocket.close();
        WriteLog("Bluetooth Closed");
}
```

6.2.0 – RavenAPI_LogTrip

```
// This is the RavenAPI call for inserting a logged trip into SQL
// this class is extending AsyncTask which means it can run
private class RavenAPI_LogTrip extends AsyncTask<Void, Void, String> {

        //define variables for later use
        private String API_URL = "http://raven-gps.com/insert.php";
        private String time_started;
        private String time_ended;
        private String driver_username;
        private String driver_reg;
        private String log_file;

        //Constructor for logTrip
        RavenAPI_LogTrip(String time_started, String time_ended, String driver_username, String
driver_reg, String log_file){
                this.time_started = time_started;
                this.time_ended = time_ended;
                this.driver_username = driver_username;
                this.driver_reg = driver_reg;
                this.log_file = log_file;
        }
```

6.2.1 - onPreExecute

```
        protected void onPreExecute() {
                // this gets executed before doInBackground and is often used for showing a loading bar
        }
```

6.2.2 - doInBackground

```
        protected String doInBackground(Void... urls) {

                try {
                        URL url = new URL(API_URL);
                        //android.util.Log.e("LOG", log_file);
                        // establishes connection to the server
                        HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                        // tells the server to use POST for the request
                        urlConnection.setRequestMethod("POST");

                        try {
                                // sets up buffers for reading from the server
                                BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                                StringBuilder stringBuilder = new StringBuilder();
```

```
                String line;

                // prepares for sending data to the server
                OutputStream os = urlConnection.getOutputStream();
                BufferedWriter writer = new BufferedWriter(
                        new OutputStreamWriter(os, "UTF-8"));
                // puts data into a json format
                JSONObject j = new JSONObject();
                j.put("time_started", time_started);
                j.put("time_ended", time_ended);
                j.put("driver_username", driver_username);
                j.put("driver_reg", driver_reg);
                j.put("log_file", log_file);
                // outputs json to log for debug purposes
                android.util.Log.e("JSON", j.toString());
                writer.write(j.toString());

                //cleans up buffer
                writer.flush();
                writer.close();
                os.close();

                //reads response from server
                while ((line = bufferedReader.readLine()) != null) {
                    stringBuilder.append(line).append("\n");
                }
                bufferedReader.close();
                return stringBuilder.toString();
            }
            finally{
                // closes connection at the end
                urlConnection.disconnect();
            }
        }
        catch(Exception e) {
            android.util.Log.e("ERROR", e.getMessage(), e);
            return null;
        }
    }
```

6.2.3 - onPostExecute

```
    protected void onPostExecute(String response) {
        // after the main call, here we can do what we want with the response
        android.util.Log.e("INFO", response);
        Toast.makeText(MainActivity.this, response, Toast.LENGTH_SHORT).show();
    }
}
}
```

PHP

7.1.0 - Check_login.php

```php
<?php
   $con = new mysqli("raven-gps.com","root","#####","raven");

   $flag=0;

   if ($con->connect_error) {
      die("Failed to connect to MySQL: " . $con->connect_error);
   }

   $sql = "SELECT * FROM logins";
   $result = $con->query($sql);

   try {
      $username = $_GET['user'];
      $password = strtoupper(hash("sha512", $_GET['pass']));
   } catch (Exception $e) {

   }

   if ($result->num_rows > 0){
      while($row = $result->fetch_assoc()){
         if ($row["username"] == $username && $row["password"] == $password){
            $flag = 1;
         }
      }
   }

   $con->close();

   echo $flag;

?>
```

8.1.0 - Insert.php

```php
<?php
   $host='raven-gps.com';
   $uname='root';
   $pwd='#####';
   $db="raven";

   $con = mysql_connect($host,$uname,$pwd) or die("connection failed");
   mysql_select_db($db,$con) or die("db selection failed");

   $flag['success']=0;

   $json = json_decode(file_get_contents('php://input'));

   try{
      $sql = "INSERT INTO trips
(`time_started`,`time_ended`,`driver_username`,`driver_reg`,`log_file`) VALUES (
         STR_TO_DATE('{$json["time_started"]}', '%m/%d/%Y-%H:%i:%s'),
         STR_TO_DATE('{$json["time_ended"]}', '%m/%d/%Y-%H:%i:%s'),
```

```
            '{$json["driver_username"]}',
            '{$json["driver_reg"]}',
            '{$json["log_file"]}')";

        $r=mysql_query($sql, $con);
    }catch (exception $e){
        echo $e;
        die(json_encode($flag));
    }

        $flag['success']=1;


    print(json_encode($flag));
    mysql_close($con);
?>
```

Arduino

9.1.0 - Raven.ino

```
#include <Arduino.h>
#include <OBD.h>
#include <SD.h>
#include <SPI.h>
#include <TFT.h>
#include <MultiLCD.h>
#include <TinyGPS.h>
#include <ArduinoJson.h>
#include "Raven.h"
#include "RavenOBD.h"
#include "images.h"

int dTime = 0;
```

9.1.1 - setup

```
void setup() {
  // put your setup code here, to run once:
  lcd.begin();
  GPS.begin(115200);
  BT.begin(9600);
  //lcd.setXY(250 - (128 / 2),150 - (128 / 2));
  //lcd.draw(RavenArudino, 128,128);
  //lcd.setXY(250 - (122 / 2),150 + (128 / 2) + 5);
  //lcd.drawBitmap(0,0, 128,128, "logo.bmp", 0,0,0);
  //lcd.image(logo, 250 - (128 / 2),150 - (128 / 2));
  //delay(500);
  obd.begin();
  delay(3000);
  lcd.clear();
  hasMEMS = obd.memsInit();
  lcd.print("MEMS:");
  lcd.println(hasMEMS ? "Yes" : "No");

#if !DEBUG_MODE
```

```
  // send some commands for testing and show response for debugging purpose
  //testOut();

  // initialize OBD-II adapter
  while(!obd.init()) {
    lcd.setCursor(0,0);
    lcd.println("Standby for OBD Connection");
  }

#endif

  char buf[64];
  if (obd.getVIN(buf, sizeof(buf))) {
      lcd.print("VIN:");
      lcd.println(buf);
  }

  unsigned int codes[6];
  byte dtcCount = obd.readDTC(codes, 6);
  if (dtcCount == 0) {
    lcd.println("No DTC");
  } else {
    lcd.print(dtcCount);
    lcd.print(" DTC:");
    for (byte n = 0; n < dtcCount; n++) {
      lcd.print(' ');
      lcd.print(codes[n], HEX);
    }
    lcd.println();
  }
  lcd.clear();
  lcd.setFontSize(FONT_SIZE_XLARGE);
  lcd.print("STATUS: ");
  lcd.setColor(RGB16_GREEN);
  lcd.println("ACTIVE");
  lcd.println();
  lcd.setColor(RGB16_WHITE);
  delay(3000);
}
```

9.1.2 - loop

```
void loop() {
  processGPS();

  int timeDiff = millis() - dTime;
  if (timeDiff > 30){
    dTime = millis();

    #if DEBUG_MODE
      rpm = rpm + 1;
    #endif

    readPIDs();
    lcd.setCursor(0,2);
    lcd.print("lat: ");
    lcd.print((float)lat / 100000, 5);
```

```
    lcd.setCursor(0,4);
    lcd.print("lng: ");
    lcd.print((float)lng / 100000, 5);
    lcd.setCursor(0,6);
    lcd.print("date: ");
    lcd.print(date);
    lcd.setCursor(0,8);
    lcd.print("time: ");
    lcd.print(time);
    lcd.setCursor(0,10);
    lcd.print("RPM: ");
    lcd.print(rpm);
    lcd.setCursor(0,12);
    lcd.print("Speed: ");
    lcd.print(_speed);
  }
}
```

10.1.0 - Raven.h

```
#define BT Serial1
#define GPS Serial2

#define DEBUG_MODE 1

LCD_R61581 lcd;
COBDI2C obd;
TinyGPS gps;
```

11.1.0 - RavenOBD.h

```
bool hasMEMS;
int deltaTime = 0;
int writeInterval = 30;

uint32_t time;
uint32_t date;
int32_t lat, lng;
int32_t rpm,_speed;
unsigned long chars;
unsigned short sentences, failed_checksum;

static const byte pidlist[] = {PID_RPM, PID_SPEED};
static const String pidNames[] = {"Rpm", "Speed"};
```

11.1.1 - testOut

```
void testOut()
{
   static const char cmds[][6] = {"ATZ\r", "ATH0\r", "ATRV\r", "0100\r", "010C\r",
"0902\r"};
   char buf[128];

   for (byte i = 0; i < sizeof(cmds) / sizeof(cmds[0]); i++) {
      const char *cmd = cmds[i];
      lcd.print("Sending ");
      lcd.println(cmd);
      if (obd.sendCommand(cmd, buf, sizeof(buf))) {
```

```
            char *p = strstr(buf, cmd);
            if (p)
                p += strlen(cmd);
            else
                p = buf;
            while (*p == '\r') p++;
            while (*p) {
                lcd.write(*p);
                if (*p == '\r' && *(p + 1) != '\r')
                    lcd.write('\n');
                p++;
            }
        } else {
            lcd.println("Timeout");
        }
        delay(250);
    }
    lcd.println();
    delay(2500);
    lcd.clear();
}
```

11.1.2 - readPIDs

```
void readPIDs(){
  int timeDiff = millis() - deltaTime;
  if (timeDiff > writeInterval){
    StaticJsonBuffer<120> jsonBuffer;
    JsonObject& json = jsonBuffer.createObject();

    json["DeltaTime"] = timeDiff - writeInterval;

#if !DEBUG_MODE

    for (byte i = 0; i < sizeof(pidlist) / sizeof(pidlist[0]); i++) {
        byte pid = pidlist[i];
        bool valid = obd.isValidPID(pid);

        if (valid) {
            int value;
            if (obd.readPID(pid, value)) {
              json[pidNames[i]] = value;
            }
        }else{
          json[pidNames[i]] = "";
        }
     }

    json["Lat"] = ((float)lat / 100000);
    json["Lng"] = ((float)lng / 100000);
    json["Date"] = date;
    json["Time"] = time;

    rpm = json["Rpm"];
    _speed = json["Speed"];
#endif
```

```
#if DEBUG_MODE

    json["Rpm"] = 800;
    json["Speed"] = 20;
    json["Lat"] = ((float)lat / 100000);
    json["Lng"] = ((float)lng / 100000);
    json["Date"] = date;
    json["Time"] = time;

#endif

    deltaTime = millis();
    json.printTo(BT);
    BT.println("");
  }
}
```

11.1.3 - processGPS

```
void processGPS(){
  while (GPS.available()){
    char c = GPS.read();

    if (gps.encode(c)){
      // retrieves +/- lat/long in 100000ths of a degree
      gps.get_position(&lat, &lng, 0);

      // time in hhmmsscc, date in ddmmyy
      gps.get_datetime(&date, &time, 0);
    }

    #if DEBUG_MODE
      _speed = _speed + 1;
    #endif
  }
}
```

SQL Script

```
USE raven;

DROP TABLE if exists raven.trips;
DROP TABLE if exists raven.logins;

CREATE TABLE IF NOT EXISTS `raven`.`logins` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `reg` VARCHAR(10) NOT NULL,
  `username` VARCHAR(32) NOT NULL,
  `password` mediumtext NOT NULL,
  PRIMARY KEY (`id`, `username`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC))
COMMENT = 'Login table for Raven-GPS authentication';

INSERT INTO logins (`reg`,`username`,`password`) VALUES (
   "BK79499",
   "rwejlgaard",
"B109F3BBBC244EB82441917ED06D618B9008DD09B3BEFD1B5E07394C706A8BB980B1D7785E5976EC049B46DF5F1326AF
5A2EA6D103FD07C95385FFAB0CACBC86"
);
```

```
INSERT INTO logins (`reg`,`username`,`password`) VALUES (
   "BE70846",
   "nwmicheelsen",
"B109F3BBBC244EB82441917ED06D618B9008DD09B3BEFD1B5E07394C706A8BB980B1D7785E5976EC049B46DF5F1326AF
5A2EA6D103FD07C95385FFAB0CACBC86"
);

CREATE TABLE `raven`.`trips` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `time_started` DATETIME NOT NULL,
  `time_ended` DATETIME NOT NULL,
  `driver_username` VARCHAR(32) NOT NULL,
  `driver_reg` VARCHAR(20) NOT NULL,
  `log_file` LONGTEXT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `reg_idx` (`driver_reg` ASC),
  INDEX `driver_username_idx` (`driver_username` ASC),
  CONSTRAINT `driver_username_FK`
    FOREIGN KEY (`driver_username`)
    REFERENCES `raven`.`logins` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
COMMENT = 'Trip table for Raven-GPS';
```