

AARHUS SCHOOL OF ENGINEERING

ANVENDTE MICROCONTROLLER SYSTEMER

6. SEMESTER

AMS projekt

Gruppemedlemmer:

Søren Landgrebe

Philip Nygaard Scmhidt

Studienr:

201508295

201506381



30. maj 2018

Indhold

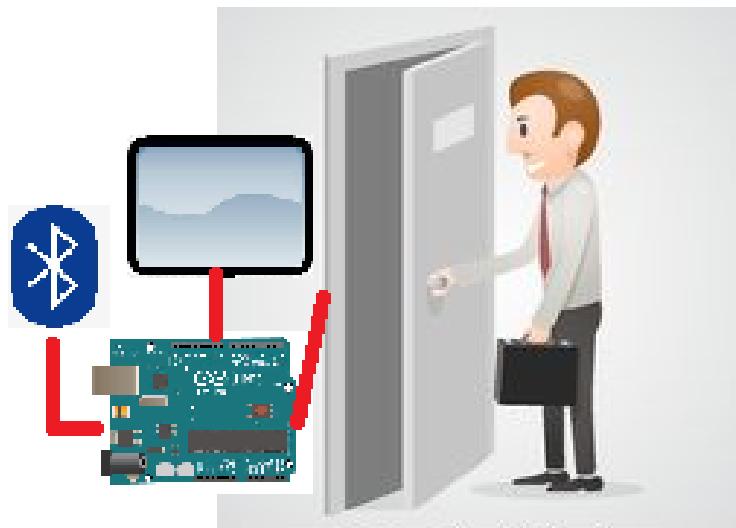
1 Indledning	3
2 Krav	4
3 Aktørbeskrivelse	4
4 Use case beskrivelse	5
4.0.1 UC 1 - ADD DEVICE	5
4.0.2 UC 2 - REMOVE DEVICE	5
4.0.3 UC 3 - LOCK ON / LOCK OFF / AUTO	6
5 System beskrivelse	7
5.1 Blokbeskrivelse BA-TA	7
5.2 Internal Block Diagram for BA-TA	8
6 System Arkitektur	9
6.1 Entity Relationship Diagram	9
7 Brugergrænseflade	13
8 Tekniske overvejelse og valg	19
8.1 Fordeler og ulemper	19
9 Tilegnelse af viden	21
9.1 Moduler, blokke og protokoller	21
9.2 Kode-klasser	21
10 Test resultater	22
11 Konklusion	23
12 Graphic display driver	25
13 Display driver	26
14 Touch driver	29
15 Bluetooth modul	30
15.1 Protokol	30
15.2 Implementering	31

Figurer

1	Konceptbillede	3
2	Aktør Kontekst diagram	4
3	Usecase diagram	5
4	Bdd af BA-TA	7
5	Ibd af BA-TA	8
6	Entity Relationship Diagrams	9
7	Sekvensdiagram for use case 1	10
8	Sekvensdiagram for use case 2	11
9	Sekvensdiagram for use case 3	12
10	welcome	13
11	Hovedmenu	14
12	Searching	14
13	Tilgængelige enheder	15
14	Valg af enhed	15
15	Fjern en enhed	16
16	Ingen enheder	16
17	Valgt fjernet en enhed	17
18	Låst op	17
19	Lås Automatisk	18
20	Bus Timing	26
21	The Dot Factory karakter	27
22	Bluetooth modulet: HC-05 GW-040.	30
23	Kommunikationseksempel med AT-kommandoer ved brug af en UART forbindelse til Bluetooth modulet.	31
24	Registrering af det returnede "OK" fra Bluetooth modulet	31
25	Al rå adresse information modtaget fra Bluetooth modulet	32
26	Brug af "delimeter" principippet ved brug af strtok_r funktionen	32

1 Indledning

Næsten alle danskere har nu til dags en smartphone med indbygget bluetooth modul, som man altid har med på sig når man forlader sit hjem. Dette vil vi gerne udnytte til at kunne gøre det nemmere for brugere, at kunne låse op og låse hoveddøren, som adskiller omverdenen fra ens dyrebare ejendele. Dette betyder at forbrugerne aldrig skal tænke mere på nøgler, da disse bliver overflødige. Dermed slipper brugeren for at skulle huske på dette, samt at skulle fumble med nøglerne når man kommer hjem med flere poser i hænderne fra dagens indkøbstur.



Figur 1: Konceptbillede

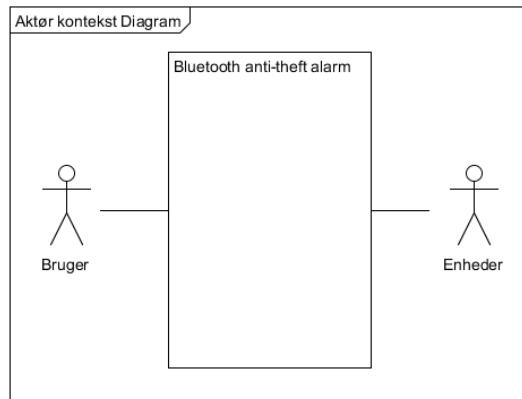
2 Krav

I dette afsnit beskrives kravene til hvilken funktionalitet systemet har.

I samarbejde med vejleder er der opstillet en række krav.

- Systemet skal kunne finde de 4 størkeste bluetooth signaler.
- Systemet skal kunne køre uafbrudt
- Systemet skal scanne efter enheder hvert 5. sekund

3 Aktørbeskrivelse

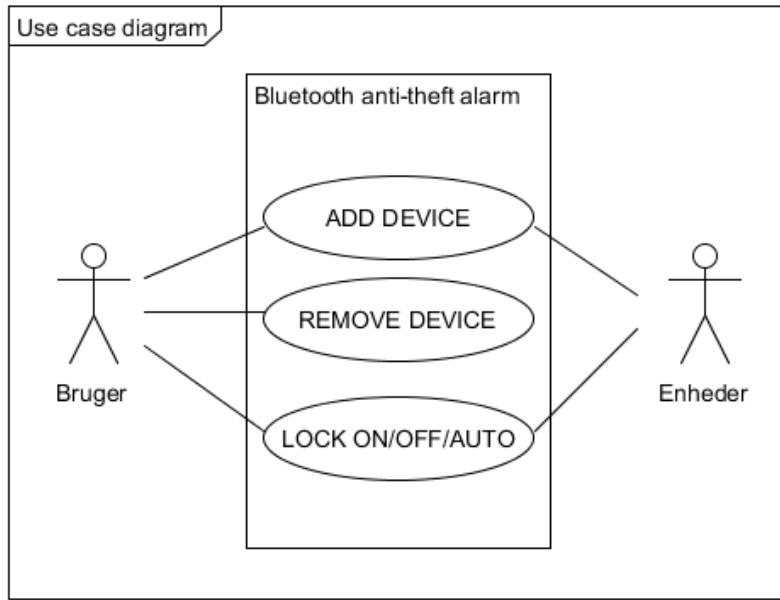


Figur 2: Aktør Kontekst diagram

På figur 2 ses aktør kontekst diagrammet som beskriver sammenhængen mellem aktørene og det system de interagere med. Aktørene er som følger:

- **Bruger:** Aktøren der interagerer med systemet og dets funktionalitet
- **BA-TA:** Bluetooth Anti-Theft Alarm systemet
- **Enheder:** Op til 4 Bluetooth enheder som er registreret af BA-TA

4 Use case beskrivelse



Figur 3: Usecase diagram

På figur 3 ses usecase diagrammet som beskriver sammenhængen mellem aktørerne og de forskellige funktionaliteter der findes for systemet.

4.0.1 UC 1 - ADD DEVICE

- Brugeren trykker på ”ENTER” når pointeren er ved ”ADD DEVICE”.
- Systemet scanner efter 4 stærkeste Bluetooth signaler
- Brugeren kan vælge imellem de registrerede bluetooth-enheder, som skal tilføjes til listen over de godkendte enheder.

4.0.2 UC 2 - REMOVE DEVICE

- Brugeren trykker på ”ENTER” når pointeren er ved ”REMOVE DEVICE”.
- Systemet præsenterer listen over de godkendte enheder
- Brugeren kan vælge imellem de godkendte bluetooth-enheder, som brugeren ønsker at fjerne fra listen.

4.0.3 UC 3 - LOCK ON / LOCK OFF / AUTO

- Brugeren trykker på ”ENTER” når pilen er ved ”LOCK ON/”LOCK OFF/”LOCK AUTO”.
- Systemet skifter state i rækkefølgen 1 - LOCK ON 2 - LOCK OFF 3 - LOCK AUTO.

5 System beskrivelse

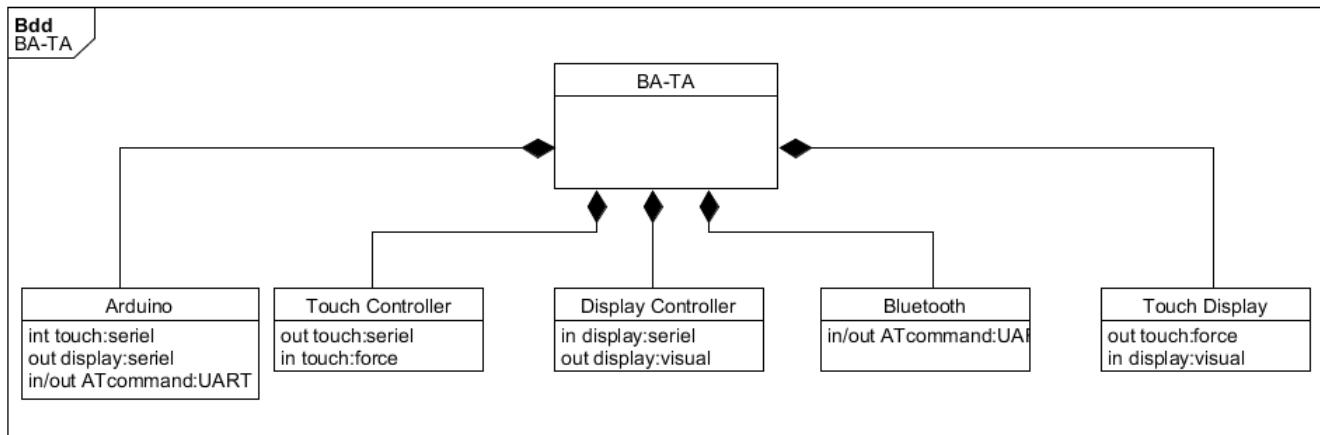
BA-TA's (Bluetooth Anti-Theft Alarm) primære funktionalitet ligger i hhv. at kunne låse op og låse hoveddøren, så snart systemet registrerer at husets ejer, og dermed ejerens smartphone, er i nærheden eller ej.

Hele systemet styres fra en samlet enhed, som kan intergeres fra brugeren gennem en skærm og dertilhørende touch funktion.

Igennem brugergrænsefladen kan brugeren både tilføje og fjerne de godkendte Bluetooth-enheder (personer) som skal kunne låse og låse op for døren.

Når alle godkendte enheder er uden for rækkevidde af systemets Bluetooth-modul, låses døren, og så snart systemet ser en godkendt enhed indenfor rækkevidden bliver døren låst op.

I under-afsnittende herunder bliver systemets blokke og de interne forbinder præsenteret. Derudover dannes der et overblik over funktionaliteten som er implementeret i projektet og systemet. På figur 4 ses et overordnet Bdd for projektet, hvor de interne forbinder forklares på figur 5.



Figur 4: Bdd af BA-TA

***Skal vi nøjes med et mere simpelt BDD hvor der eksempelvis ikke står nogen til Arduinoen og de andre?

5.1 Blokbeskrivelse BA-TA

I det følgende vil komme en beskrivelse af de enkelte blokke i BA-TA og deres interne funktionalitet.

Arduino

Arduino'en, mega2560, initialiserer og styrer alt funktionaliteten som indgår i systemet. Ar-

duino blokken står for at initialisere Touch Display, Touch Controller og Display Controller, som bruges til at kunne modtage signaler fra Touch-delen, samt at sende signaler til Display'et og dermed få repræsenteret noget til brugeren. Yderligere bruges Arduinoen til at kommunikere med Bluetooth-blokken.

Touch Controller

Touch Controller står for at modtage værdier fra Touch Display, samt at sende den modtagede værdi videre til Arduino'en.

Display Controller

Display Controller modtager værdier fra Arduino'en og sende sende værdien videre til Touch Display.

Bluetooth

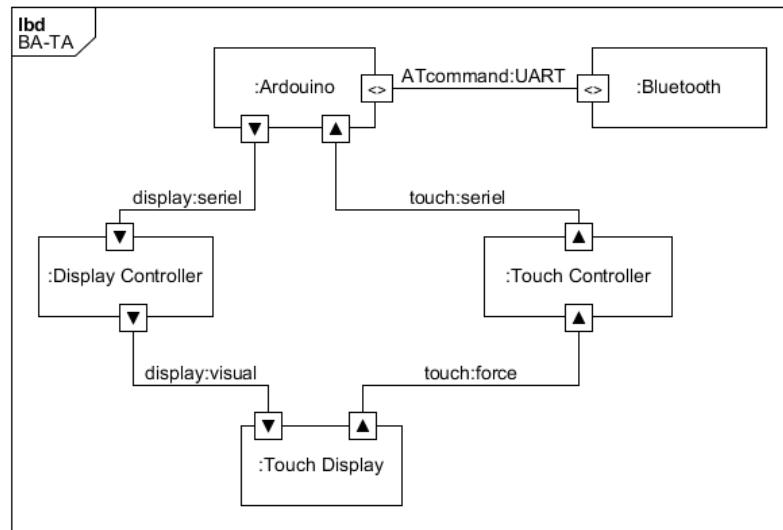
Bluetooth modul, HC-05, som kommunikkerer med Arduino'en vha. AT-kommandoer.

Touch Display

Touch Display fungerer som brugergrænseflade og er dermed måden brugeren kan interagere med systemet.

5.2 Internal Block Diagram for BA-TA

På figur 5 ses et overordnet IBD over selve systemet, som er bygget på Bdd'et.



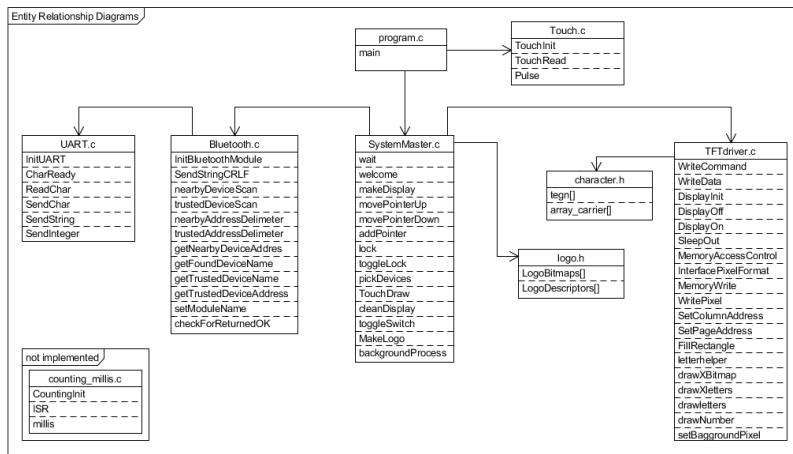
Figur 5: Ibd af BA-TA

6 System Arkitektur

Systemarkitekturen i projektet har været en iterativ proces og er derfor blevet rekonstrueret og ændret undervejs i projekt-forløbet. Arkitekturen i systemet danner rammer for, hvordan BA-TA's logiske system er opbygget og implementeret. Hermed dannes der et overblik over den logiske funktionalitet og hvordan den bruges på tværs af de forskellige drivere i systemet.

6.1 Entity Relationship Diagram

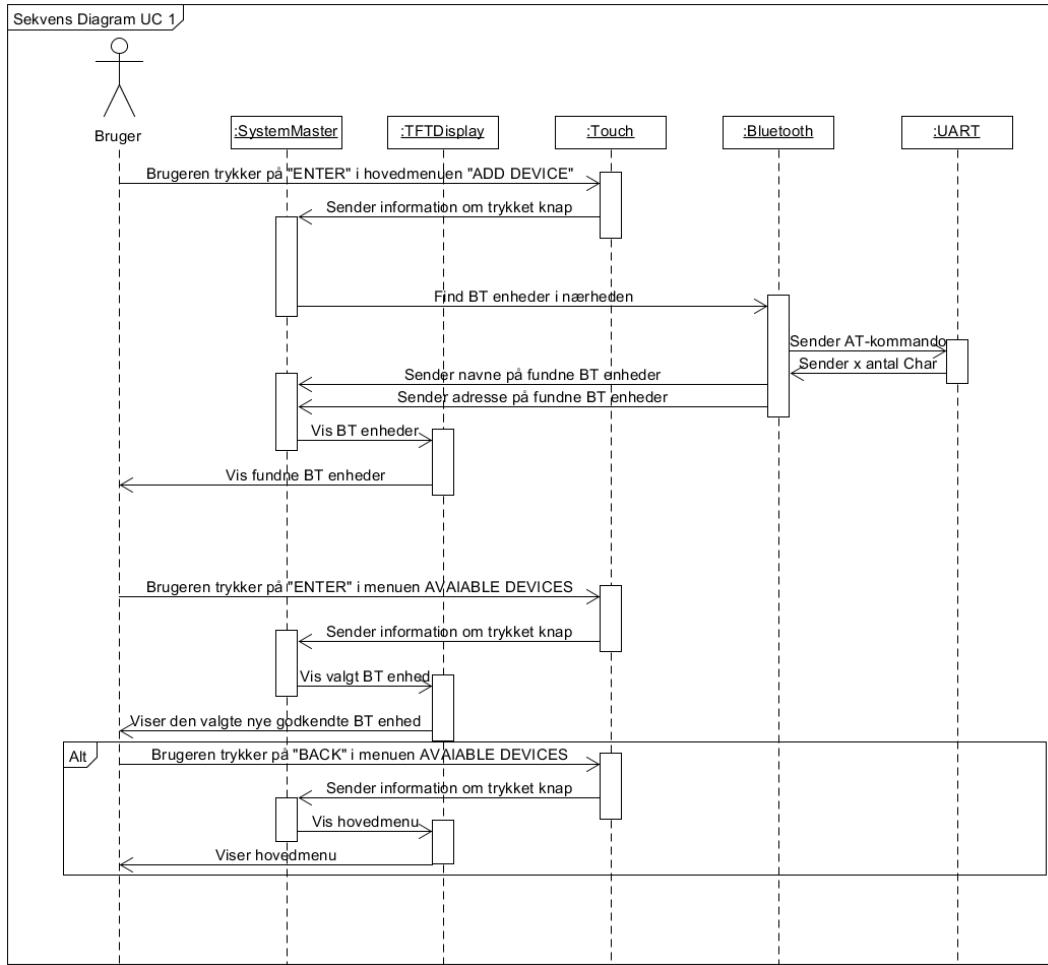
På nedenstående figur, 6, danner der et overblik over den logiske funktionalitet på tværs af systemet, som er udmundet i funktioner i de forskellige driver-klasser.



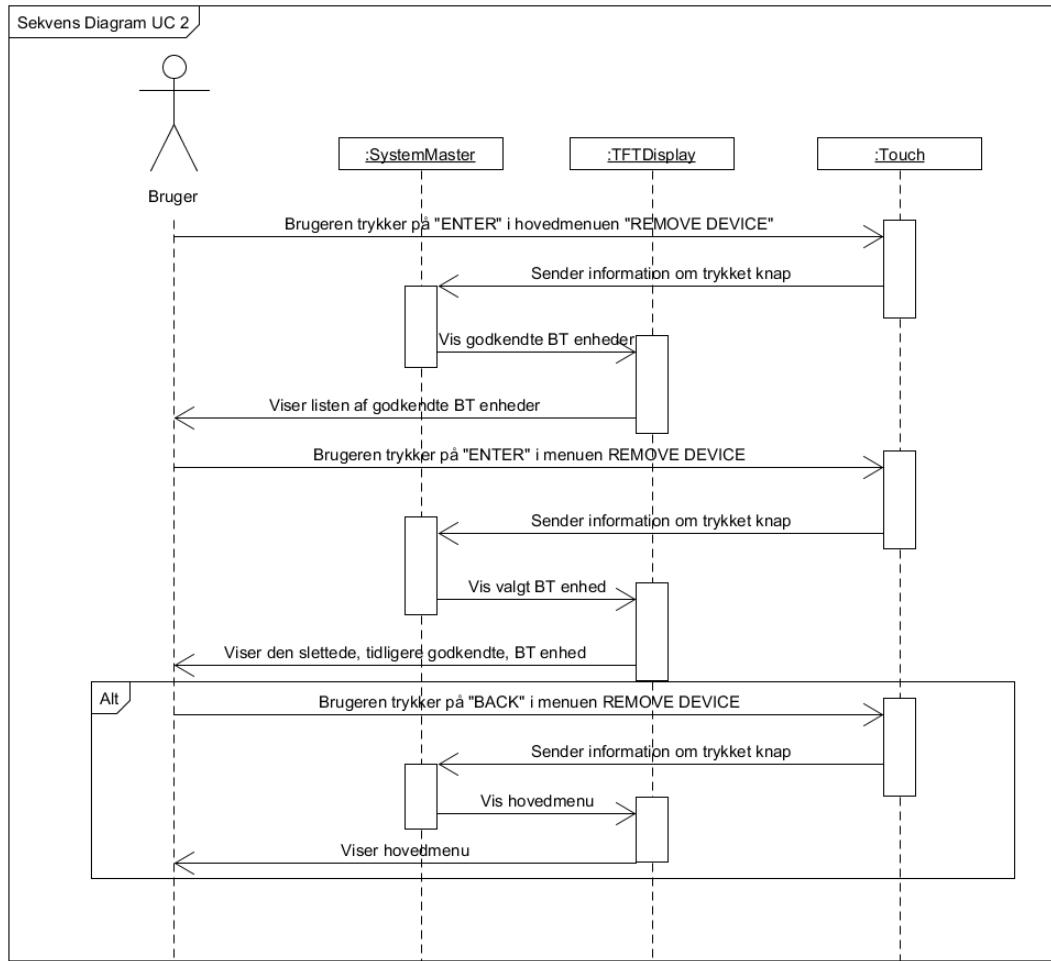
Figur 6: Entity Relationship Diagrams

6.2 Sekvensdiagrammer

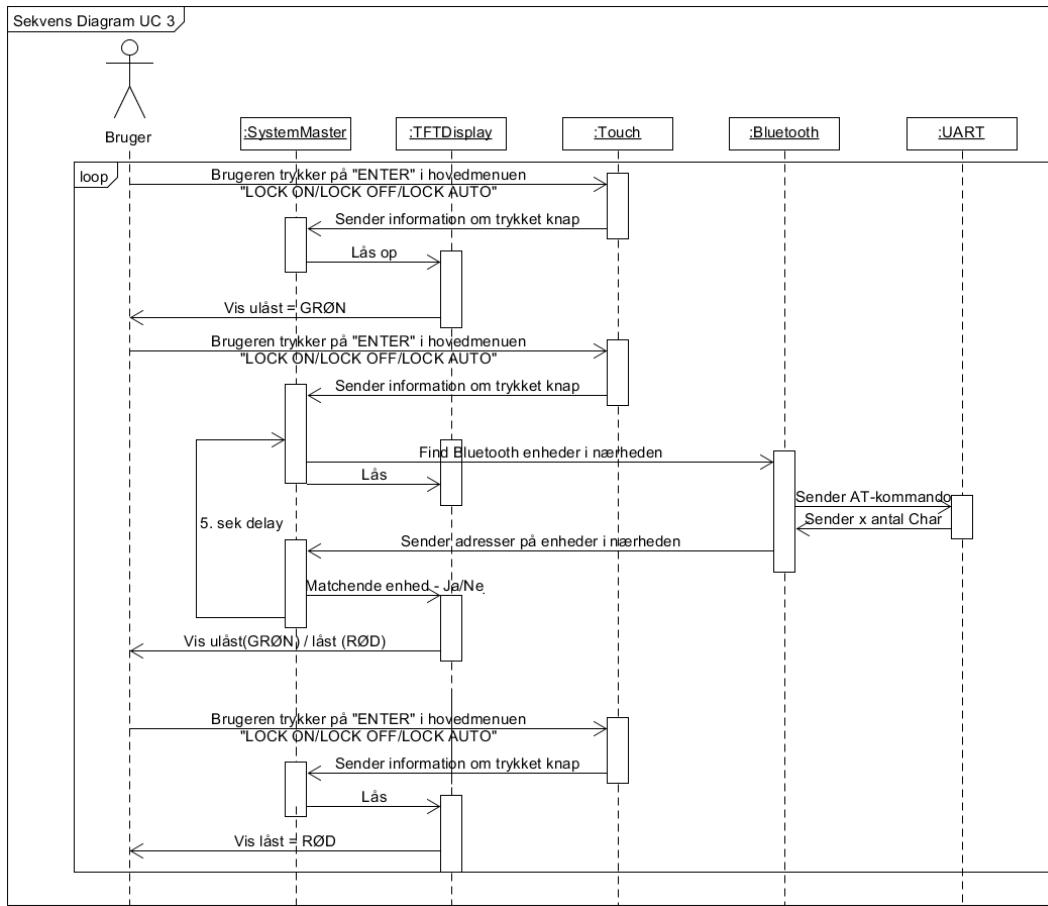
På nedenstående figur, 6, danner der et overblik over den logiske funktionalitet på tværs af systemet, som er udmundet i funktioner i de forskellige driver-klasser.



Figur 7: Sekvensdiagram for use case 1



Figur 8: Sekvensdiagram for use case 2



Figur 9: Sekvensdiagram for use case 3

7 Brugergrænseflade

I dette afsnit gives der et overblik over brugergrænsefladen og brugerens mulighed for interaktion med denne. Afsnittet er bygget op af billeder, hvor hvert billede viser brugergrænsefladens visuelle struktur.

Første display man møder er velkommen. Her går Arduino'en igang med at initialisere de forskellige moduler, således de er klar til brug.



Figur 10: welcome

Næste display er hovedmenuen for systemet, hvor der er tre forskellige muligheder, som ses på

figur 11. Yderligere ses der 4 trykknapper, hvor brugeren kan interagere med systemet. Med "Enter" vælger brugeren en funktion. "Back" gør brugeren i stand til at gå tilbage til hovedmenuen, hvis man forinden har trykket "ENTER" ved "ADD DEVICE" eller "REMOVE DEVICE". "Up" og "Down" flytter pilen, hhv. op og ned. Herudover ses der en farvet boks øverst i højre hjørne, som indikerer låsens tilstand. Denne lås viser grøn ved låst op og rød ved låst. Øverst til venstre på skærmen vises projektets logo.



Figur 11: Hovedmenu

Hvis brugeren trykker på ”ADD DEVICE” søger BA-TA efter de 4 størkeste Bluetooth signaler, som er indenfor rækkevidde.



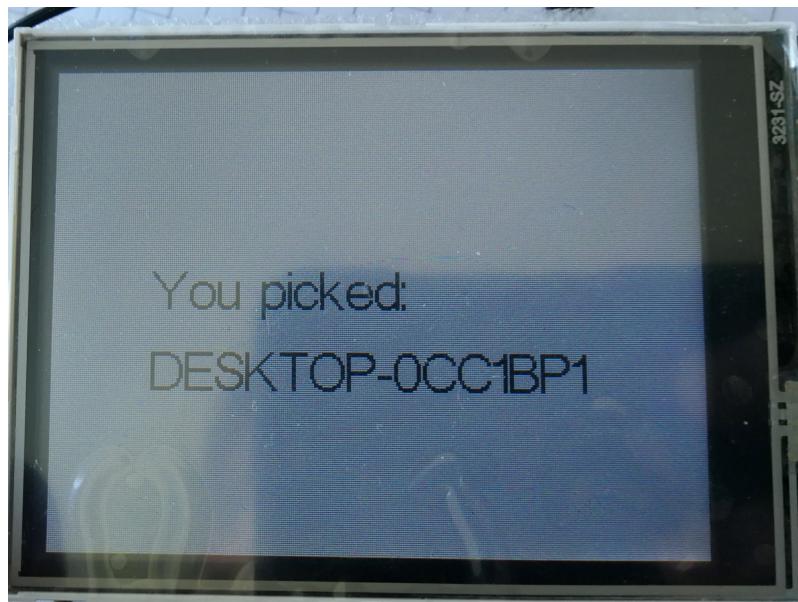
Figur 12: Searching

Herefter vises de fundne Bluetooth-enheder, som systemets Bluetooth-modul har fundet. Dette ses på figur [13](#).



Figur 13: Tilgængelige enheder

Hvis der vælges en enhed ved et tryk på ”Enter”, så vises navnet på den valgte enhed. Herefter går skærmens tilstand tilbage til hovedmenuen.

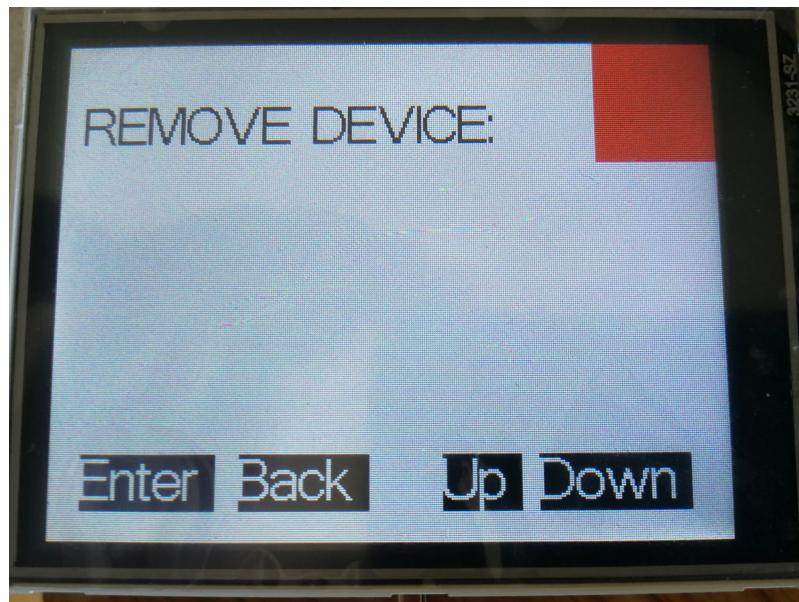


Figur 14: Valg af enhed

Hvis brugeren herefter vælger funktionen ”REMOVE DEVICE” fra figur 11, åbnes der en liste over de godkendte Bluetooth-enheder, der er gemt Dette ses på 15.. Hvis der på forhånd ikke er nogle gemte godkendte Bluetooth-enheder og listen dermed er tom, så vil displayet også være tomt, som vist på figur 16.



Figur 15: Fjern en enhed



Figur 16: Ingen eneheder

Såfremt der er godkendte Bluetooth-enheder på listen, kan brugeren vælge at slette en bestemt enhed. Dette sker ved at trykke "Enter" ved enheden, som vælges ved hjælp af "Up" og "Down" og indikeres af pilen. Displayet viser på figur 17 en besked om hvilken enhed brugeren har valgt at slette.



Figur 17: Valgt fjernet en enhed

Hvis brugeren vælger den sidste mulighed/state i hovedmenuen, ”LOCK OFF”, låses systemet op, teksten skifter og låsen skifter farve. Dette ses på 18.



Figur 18: Låst op

Hvis der endnu engang trykkes på den sidste state i hovedmenuen, ændres teksten, og låsen skifter, alt efter BA-TA's funktionalitet. Hvis der trykkes endnu engang på sidste state, ændres låsen til default, hvilket er låst.



Figur 19: Lås Automatisk

8 Tekniske overvejelse og valg

8.1 Fordeler og ulemper

Da projektets omfang og opbygning har været meget frit, er der også blevet lavet nogle valgt og fravælg i process fasen. Dette vil vi give et overblik over i dette afsnit, forklare fordele og ulmepær ved hvert modul vi har valgt, og driverne dertil. Herunder gives et overblik over de emner gruppen mener har været mest essentielle ift fordele og ulemper af de forekellige moduler.

TFT Display

Da omfanget for projektet inkluderede et display, gjorde gruppen et valg om at bruge TJC-9341-032 som display, og [ILI9341](#) som display controller. Dette [ILI9341](#) er valgt på baggrund af, gruppen har abejdet med dette display modul tidligere, derudover var dette også tilgængeligt. Til selve display'et er der valgt kun at kunne skrive til display'et og ikke læse fra det. Fordelen ved kun at sende data til displayet, er at det gør opsætningen meget nemmere for udvikleren.

Fordelen ved at initialisere driveren til at kunne modtage data fra displayet, ville være at programstrukturen, kunne spørge skærmen hvilket display, der var på skærmen nu. Dette vil sikre at programstrukturen altid ved hvilket frame der vises på displayet. Ulempen ved den måde gruppen har initialiseret driveren på, er at microcontroller skal holde styr på hvilken frame, der vises på skærmen. Dette gør det mere udfordrende for udvikleren, og derved gør at koden kommer til at fylde mere. Hvis driveren skulle laves på en memory kritisk microcontroller, ville det være en fordel at tilføje skærmen at kunne læse hvilket frame der står på skærmen.

Vi har ikke brugt meget tid på at undersøge alternative display. Der blev valgt at et Alphanumeric display, som tidligere var bearbejdet, ikke ville opfylde de krav vi havde til displayet, og da controlleren shielded ITDB02, havde en Touch controller del, blev Alphanumeric display'et valgt fra.

Gør brug af globale variabler

Igennem de forskellige drivere gøres der brug af globale variabler. Dette er gjort for at simplificere udviklingen af driverne. Denne metode at programmere på er dog ikke god programmerings-skik, og hvis gruppen havde afsat mere tid til udviklingen, ville det første være at erstatte disse globale variabler, med get() og set() metoder.

Bluetooth Modul

De første uger af udviklingsperioden var der i gruppen større problemer med det Bluetooth modul der først var blevet lånt af [Embedded Stock](#) til implementering i systemet. Efter en uges tid med troubleshooting på hvorfor den ikke kunne sættes op til at søge efter andre enheder, vidste det sig at det udleverede modul ikke var det rigtige modul som gruppen havde bestilt. Der var blevet bestilt et HC-05 modul, men istedet udleveret et HC-06 modul. Forskellen kan ikke ses med det blotte øje, da modulerne fysisk er ens og den eneste reelle forskel er firmwaren den er blevet sendt afsted med fra fabrikken. HC-06 firmwaren

er meget mere simpelt bygget op og understøtter derfor ikke samme antal avancerede AT-kommandoer. Dermed kan den ikke initialiseres til at kunne søge efter andre enheder og dermed lave inquiries.

Herefter blev der lånt et anerledes HC-05 med et større breakout-board, hvilket efter nogle få timers debugging og troubleshooting måtte konkluderes at være i stykker. Det lykkedes os dog igennem Embedded Stock at få lånt et nyt HC-05 modul af anerledes breakout-board. Dette var funktionelt og endelig kunne gruppen begynde på selve implementeringen af Bluetooth-modulet i systemet, som inkluderer alle de funktioner den skal kunne understøtte fra Arduino'en. Der var dog også snak om hvorvidt gruppen skulle bestille et dyrere Bluetooth-modul på Amazon, men i og med vi først fik prøvet det andet af, kunne gruppen konstatere at det fint ville fungerere i sammenhæng med resten af systemet. Yderlige havde gruppen på dette tidspunkt ikke behov for flere forsinkelser med Bluetooth-moduler og derfor var der ikke incitament for at skulle vente på at posten ville nå frem.

Den benyttede Bluetooth-modul i systemet, som kan ses på figur 22, er dog anderledes bygget op på breakout-boardet, hvilket betyder, at der har skulle træffes nogle valg alt efter, hvordan gruppen ville bruge Bluetooth-modulet i systemet. Bluetooth-modulet har 2 muligheder: den kan styres af kommandoer igennem Bluetooth-protokolen OTA (over-the-air) af enheder den har oprettet en Bluetooth forbindelse til og så kan man styre den ved brug af UART forbindelse og AT-kommandoer til dens RX/TX ben.

9 Tilegnelse af viden

Kort beskrivelse af hvordan vi har tilegnet og den viden der har været nødvendig for at projektet er i den version den er i på nuværende tidspunkt, samt den viden der ligger bag funktionaliteten i de forskellige blokke/moduler.

9.1 Moduler, blokke og protokoller

AT-kommandoer

Dele af gruppen har på forhånd været bekendt med AT-kommandoer og deres opbygning og struktur. Denne viden er tilegnet i praktik på 5. semester, hvor der fra en microcontroller blev brugt AT-kommandoer til at initialisere og styre et IoT-device.

Bluetooth modulet - HC-05

Tilegnelsen af viden til Bluetooth-SMT-modulet (Surface-Mount-Technology), HC-05, har forummer og generelle internetsøgninger været nødvendige. Da modulet ikke bliver produceret af én bestemt producent har der ikke været et decideret datasheet som gruppen har kunne bruge. Derimod er modulet meget kendt og er brugt af mange mennesker. Det afspejler sig i de mange forskellige blogge gruppen har fundet, hvor modulet er blevet brugt i et projekt. Her har vi kunne hente nok information om modulet og de AT-kommandoer som modulet kan genkende. Især nedenstående side har været meget brugbar: <http://www.martyncurrey.com/arduino-with-hc-05-bluetooth-module-at-mode/> (Lav til referencer senere)

9.2 Kode-klasser

Her kan evt tilføjes noget om funktioner eller andre principper som bliver brugt i koden, som er viden vi har tilegnet os undervejs

*** delimetering i Bluetooth.c streat strcpy strdup ***

10 Test resultater

Her skal der smides alle test resultater ind af systemet. Tag brug af Use Cases som udgangspunkt til disse. Dermed er der også en bedre rød tråd i dokumentet og dermed grundlag for en god konklusion efterfølgende.

11 Konklusion

Sørg for ordentlig rød tråd med indledning, hvad systemet kan af funktionalitet på nuværende tidspunkt, udfordringer, valg, tilegnelse af viden og end ud i hvor høj en gråd projekt-målene er blevet mødt.

References

- [1] Gan and Kuo.
Embedded Signal Processing with the Micro Signal Architecture, Chapter 4.4.1
John Wiley 1st Ed. 2007.
- [2] Gan and Kuo.
Embedded Signal Processing with the Micro Signal Architecture, Chapter 7.2.2.1
John Wiley 1st Ed. 2007.

12 Graphic display driver

Som brugergrænseflade i dette projekt bruges et ITDB02, som er valgt da der tidligere er arbejdet med netop dette produkt. Dertil kommer ILI9341 som driver til selve display'et.

Driver softwaren til hele displayet, er delt op i flere forskellige cpp filer, dette er gjort for at gøre koden mere overskuelig, og gøre funktionaliteten mere effektiv. Herunder forsøges at gøre et overblik over de forskellige cpp filer og deres integeren.

—Skriv noget når det hele er færdig her

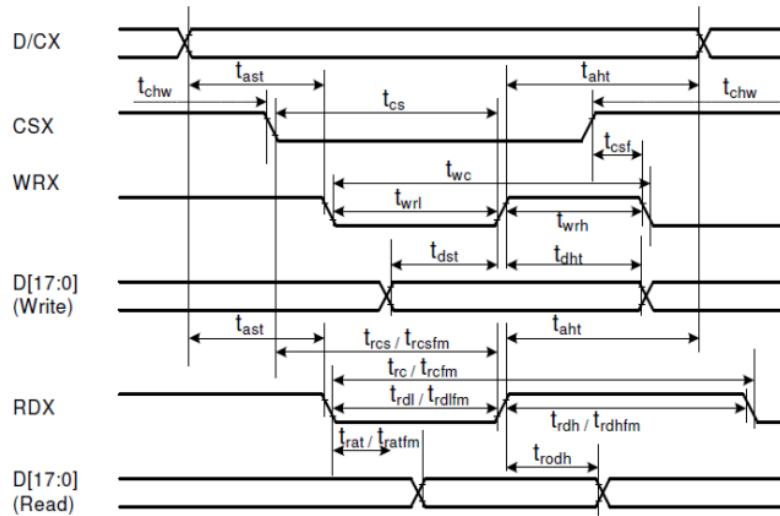
13 Display driver

Igennem øvelser til undervisningen, er der blevet opbygget en driver til et grafisk display. Igennem processen til dette projekt er der blevet modifieret og arbejdet videre på selv samme driver. Cpp filen ligger under TFTdriver, som er bygget op af flere forskellige funktioner. I dette afsnit vil der gives et overblik over de mest essentielle funktioner. For at få en forståelse af selve opbygningen af driveren, henvises til datasheet for controlleren. [ILI9341](#)

For at opbygge en basisforståelse for driveren, vil der først blive forklaret `DisplayInit()`. Først sætter vi de porte vi skal bruge fra Arduino til hhv. indgange og udgange. Vi har dog valgt ikke at have tilbagemeldinger, og derfor er der ikke initialiseret nogle indgange. Herefter sætter vi RESET, CS, WRX, RDX høje ift figur 20. Der bliver kørt en reset (tjek lige koden med timing, burde være kortere), for at resette displayet, og bagefter sendt fire kommandoer, som kan findes i command list i databladet. Sleep Out, Display On, Pixel format set = 16 bit og Memory Access Control (BGR = 1) (Tjek ift henning hvad dette gør).

18.3 AC Characteristics

18.3.1 Display Parallel 18/16/9/8-bit Interface Timing Characteristics (8080- I system)



Figur 20: Bus Timing

Herunder vil de essentielle funktioner deles op i hvert sin tabel. Flere af funktionerne gør brug af både `Writecommand()` og `Writedata()`, som er opbygget udfra Bus Timing figur 20. Derudover bruger flere af funktionerne `SetColumnAddress()` og `SetPageAddress()`, som er bygget op fra datasheet 8.2.20. og 8.2.21.:

Included Font:

character.h
Karakter størrelse: 24*24 pixels
Antal karakter: 95



Figur 21: The Dot Factory karakter

Funktions:

FillRectangle(StartX, StartY, Width, Height, Red, Green, Blue)
Funktion som laver en firkant med den valgte baggrundsfarve
Paramentre: StartX: Startværdi på x-aksen StartY: Startværdi på y-aksen Width: Bredden på firkanten Height: Højde på firkanten Red: farve(værdi) Green: farve(værdi) Blue: farve(værdi)

FillRectangle er en funktion som visuelt kan ændre bits til en anden farve på den givne skærm. Fra input vælges startx og starty samt længde og bredde. Derefter laves en ud fra de input der er givet. Farven afhænger også af inputtet. Det der reelt sker, er at den givne farve bliver skrevet til hver eneste pixel, i det sted der er valgt en firkant.

drawletters(str[], startx, starty, Red, Green, Blue)
Funktion som modtager en streng, og konverterer ascii værdien til den rette værdi ift character.h
Paramentre: str[]: Modtager en streng Startx: Startværdi på x-aksen Starty: Startværdi på y-aksen Red: farve(værdi) Green: farve(værdi) Blue: farve(værdi)

drawletters er en startfunktion til drawXletter, i denne funktion, findes længden på den på det givne tegn, som skal skrives. Herudover findes lægden at det givne loop drawXletter skal fortsætte, ift hvor mange bytes der hører til det givne tegn. Denne funktion står for at finde de respektive værdier fra array_carrier, og sende dem med til drawXletter. til sidst sættes

den nye start_x, hvor næste tegn kan placeres, her er der valgt et mellemrum fra udvikleren på 1 pixel.

drawXletter(bitmap[],length,count,startx,starty, letter, modulus,Red, Green, Blue)
--

Funktion som står for at skrive til hver bit, med værdier fra letterhelper()
--

Paramentre:

bitmap[]: Henter en byte fra character.h
--

length: Fortæller funktionen, hvor bred karakteren der skal skrives er
--

count: hvor mange bytes funktionen skal køre igennem for at lave hele karakteren
--

Startx: Startværdi på x-aksen

Starty: Startværdi på y-aksen

Red: farve(værdi)

Green: farve(værdi)

Blue: farve(værdi)

Note: Funktionen sletter gamle karakterer, da en ny smallere karakter end forrige stadig vil forblive på displayet

drawXletters står for at farve hver pixel, så der laves det tegn som er anmodet, dette sker ved at funktionen modtager information fra drawletters, som har information om tegn[], som er hentet fra character.h. Dette array, består af et bitmap af alle ascii tegn, og byte værdier som tegner hver pixel. Igennem funktionen sikres at bredden på tegnet overholdes, og et nyt startx punkt bliver sat, samtidig med at starty (colum) sættes til næste linje. dette gøres indtil den del af arrayet er løbet igennem som er givet fra drawletters. Til sidst i koden slettes (pixels = hvid) tegn eller pixels, som ligger forand det nye tegn som er tegnet. Dette sker for at sikre at tegnene ikke flyder ind over hinanden, hvis der findes gamle tegn på pladsen, hvor den nye tegnes.

14 Touch driver

Da projektet skulle bruge integering af en bruger, er der valgt at inkudere en touch driver som gør brug af **XPT2046** Touch Screen Controller, som allerede var en del af **ITDB02** Arduino MEGA shield, som bliver også brugt i Display driveren.

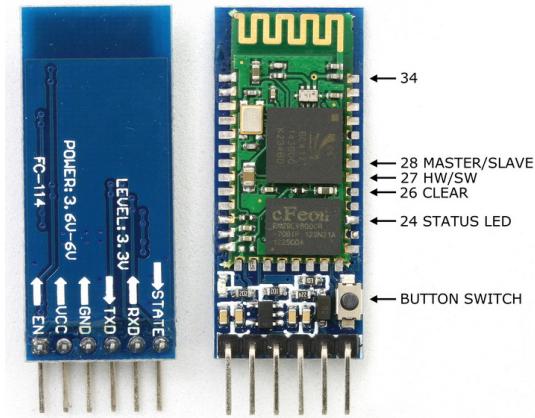
Driveren har tre funktioner, hvor Init() sætter de forskellige porte til hhv indgange og udgange, og derefter sætter de respektive ben til enten høj og lav. Funktionen pulse() står for at lave en puls på clk benet som er opsat ift. Figur 15 i **XPT2046** datasheet. Herunder vil der laves en tabel over den sidste funktion.

TouchRead(xy)
Funktion som står for at læse værdien fra brugerinputtet.
Paramentre:
xy: Styrer om retur værdien skal være for x eller y
Retur: Værdien for enten x eller y.

TouchRead står for at læse den værdi som bliver sendt over en seriel linje fra touch controlleren. Denne værdi skal ligges korrekt ind i en int, og bliver lagt en til hver bit, hvor gang der modtages ettal på PINE bit 5, herefter returneres int værdien.

15 Bluetooth modul

Bluetooth-modulet der bliver brugt i BA-TA systemet er af typen HC-05. Denne type-version understøtter nemlig mulighed for at tage rollen som både Master/Slave i et Bluetooth Master/Slave forhold imellem to parrende Bluetooth-enheden. Bluetooth modulet er sammensat med et breakout- board i bunden af modulet, hvilket giver nem adgang til de brugbare pins på selve Bluetooth-modulet, hvilket også kan ses på figur 22



Figur 22: Bluetooth modulet: HC-05 GW-040.

15.1 Protokol

Der kommunikeres til systemets Bluetooth-modul igennem en UART-forbindelse. Gruppens implementerede UART-driver på Arduino'en gør det muligt, at Arduinoen og Bluetooth-modulet kan kommunikere sammen vha. AT-kommandoer. AT-kommandoerne (Attention-commands) som Bluetooth-modulet understøtter kan bruges til generel opsætning af modulet, sætte Master/Slave roller, scanne for nærtliggende Bluetooth-enheder, samt at spørge om navn på fundne Bluetooth-enheders adresser. AT-kommandoerne er en general standard som bruges til at sende instruktioner til et modul og dermed er de opbygget på samme måde for hvert modul der bruger dem. På en etableret UART forbindelse vil man kunne sende en "AT+kommando" til modulet, hvorefter modulet vil udføre kommandoen, og evt sende et svar tilbage, samt et OK, hvilket indikerer at beskeder var succesfuldt modtaget af modulet. Det samme er gældende for HC-05 Bluetooth modulet som indgår i BA-TA systemet. Eksempler på brugte AT-kommandoer til Bluetooth-modulet i BA-TA:

- "AT+INIT" – Initialisering af Bluetooth-modulets Bluetooth SPP profil
- "AT+INQM,0,5,4" – Sætter scanner parameter. I dette tilfælde at der maksimum skal findes 5 Bluetooth enheder og at timeout på en scanning af Bluetooth enheder er $4 * 1.28$ sekunder.
- "AT+RNAME?jndsæt,Adresse,Her;" – Bluetooth-modulet spørger om navnet på adressen til den respektive indsætte adresse.

En succesfuld AT-kommando vil altid få Bluetooth-modulet til at returnere et OK. Såfremt der eksempelvis bliver spurgt om et navn på en respektiv Bluetooth-enheds adresse, så vil Bluetooth-modulet også returnere den fundne Bluetooth-enheds navn og et OK til sidst.

```

AT+INQ
+INQ: 6CB7:49:5B276A,5A020C,7FFF
+INQ: 74E5:43:96BCDF,A010C,7FFF
OK
AT+RNAME?74E5,43,96BCDF
+RNAME:DESKTOP-0CC1BP1
OK

```

Figur 23: Kommunikationseksempel med AT-kommandoer ved brug af en UART forbindelse til Bluetooth modulet.

På ovenstående figur, 23, kan vi se et eksempel på kommunikation vha. en UART forbindelse til Bluetooth modulet. I eksemplet bliver der sendt et ”AT+INQ” til Bluetooth-modulet. Denne besked betyder inquiry, og derfor går Bluetooth modulet straks igang med at undersøge hvilke nærtliggende Bluetooth enheder den kan finde. Herefter svarer den tilbage med et ”+INQ;:AdressePåFundetEnhed;” hver gang den har fundet en ny enhed. Dette forsætter den med indtil den når sit maks antal devices eller timeout grænse, som er bestemt af kommandoen ”AT+INQM”. Når timeout eller maks antal enheder er nået returnerer Bluetooth modulet også et OK for at indikere at den er færdig. Herefter er den også klar igen til at modtage nye beskeder.

15.2 Implementering

Da AT-kommandoerne og retur-beskederne er opbygget med samme struktur, har gruppen med fordel taget brug af dette til, at kunne udvikle en Bluetooth-kommunikations-driver, som netop udnytter denne gentagende og genkendelige proces.

```

for(i = 0; i < 300; i++)
{
    neabyRawAddresses[i] = ReadChar();

    if(neabyRawAddresses[i-2] == 'O' && neabyRawAddresses[i-1] == 'K' && neabyRawAddresses[i] == '\r')
    {
        //SendStringCRLF("OK\r was registered - now breaking for-loop");
        break;
    }
}

```

Figur 24: Registrering af det returnede ”OK” fra Bluetooth modulet

På figur 24 kan vi se et eksempel på, hvordan gruppen har udnyttet at der altid vil blive sendt et OK retur til sidst, når Bluetooth modulet indikerer at den er færdig med den instruktionen/kommandoen som den har modtaget forinden. I eksemplet fra figuren benytter

gruppen sig af dette til at tjekke hvornår Bluetooth modulet har sendt alt information om Bluetooth enheders adresser, som den har registreret ved sit "AT+INQ".

```
Raw address information was received:
{h0A}E{h0A}+INQ:74E5:43:96BCDF,A010C,7FFF
+INQ:C0EE:FB:DAF25E,5A020C,7FFF
+INQ:E4A7:A0:D8F42C,2A010C,7FFF
+INQ:6CB7:49:5B276A,5A020C,7FFF
OK{h0D}*8å€{h7F}iyie{hFFFFFFFFFFFFF8E}=if{h7F}{hFFFFFFFFFFF97}~{{h1D}{h7F}}{hFFFFFFFFFFF9F}
Bub{hFFFFFFFFFFFFF97}?{hFFFFFFFFFFFFF8E}
```

Figur 25: Al rå adresse information modtaget fra Bluetooth modulet

Herefter kan der analyseres på det char array som alt adresse information for alle fundne enheder ligger i. Formålet med dette er at få skilt informationen ad således vi tage separere de forskellige adresser fra hinanden. Dette skyldes at dataen i det pågældende char array højest sandsynligt er loaded med flere adresser, som det eksempelvis kan ses på figur 25

Næste skridt er hermed at skille alt den information vi får udeover MAC-adressen til de bestemte enheder. Dette gøres ved brug af "delimeter" principippet. Delimeter principippet kan bruges i dette tilfælde, da alle modtagede adresser i char array'et er sammensat og seperaret fra hinanden på en kontinuerlig måde. Hver MAC-adresse starter efter "+INQ:", de 3 dele af MAC adressen er separeret af ":" og MAC-adressen slutter ved et ",". Dette kan også ses på figur 25.

```
//Splitting (delimetering) the MAC addresses into 3 parts
char *delimeterColon = ":";
char *delimeterComma = ",";

char *nearbyHeader1 = strtok_r(nearbyRawAddress, delimeterColon, &nearbySaveptr1);

int q = 0;

//Adresse eksempel:
//{h0A}E{h0A}+INQ:74E5:43:96BCDF,A010C,7FFF
for(q = 0; q < numberOfNearbyAddresses; q++)
{
    if(q >= 1)
    {
        nearbyHeader1 = strtok_r(NULL,delimeterComma, &nearbySaveptr1);
        nearbyHeader1 = strtok_r(NULL,delimeterColon, &nearbySaveptr1);
    }

    nearbydataArray[1] = strtok_r(NULL, delimeterColon, &nearbySaveptr1); //Indeholder første del af adressen = 74E5
    nearbydataArray[2] = strtok_r(NULL, delimeterColon, &nearbySaveptr1); //Indeholder anden del af adressen = 43
    nearbydataArray[3] = strtok_r(NULL, delimeterComma, &nearbySaveptr1); //Indeholder tredje del af adressen = 96BCDF
```

Figur 26: Brug af "delimeter" principippet ved brug af strtok_r funktionen

På ovenstående figur, 26, kan vi se det "delimeter-principippet" som er taget i brug for at kunne skære alt andet end de 3 dele af adressen (i dette tilfælde, 74E5, 43 og 96BCDF).

Herefter skal adressen nemlig sammensættes på en ny måde, således at vi kan få Bluetooth modulet til at spørge adressens respektive Bluetooth enhed om enhedens navn. Dette er en

nødvendighed for at kunne vise navnet til brugeren i brugergrænsefladen i use case 1 og use case 2.

Før at Bluetooth modulet kan registrere at systemet forespørger navnet på en fundet Bluetooth enhed, og dens respektive adresse, skal det dermed følge denne form:

”AT+RNAME?74E5:43:96BCDF” og ikke:

”AT+RNAME?74E5:43:96BCDF”, som vi modtog adressen i første omgang fra Bluetooth modulet. Forskellen er ikke stor, men udbytningen af koloner til kommaer er forskellen på om Bluetooth modulet kan genkende og eksekvere kommandoen den modtager. Dermed har der været et implementeringsbehov for at kunne tage højde for dette. Dette tager resten af koden i funktionerne addressDelimeter og trustedAddressDelimeter sig af. Forskellen på disse to funktioner er at den ene bruges til at gøre det for de enheder som skal på den godkendte liste over Bluetooth enheder i systemet. addressDelimeter bruges derimod til når låse staten er sat til auto, da funktionaliteten i de 2 funktioner varierer i forhold til hinanden.

***** HVIS DER ER PLADS TIL MERE TEKST KAN NAVNE-FUNKTIONEN GODT UDDYBES HER**