

AARHUS SCHOOL OF ENGINEERING

ANVENDTE MICROCONTROLLER SYSTEMER

6. SEMESTER

AMS projekt

Gruppemedlemmer:

Søren Landgrebe

Philip Nygaard Scmhidt

Studienr:

201508295

201506381



30. maj 2018

Indhold

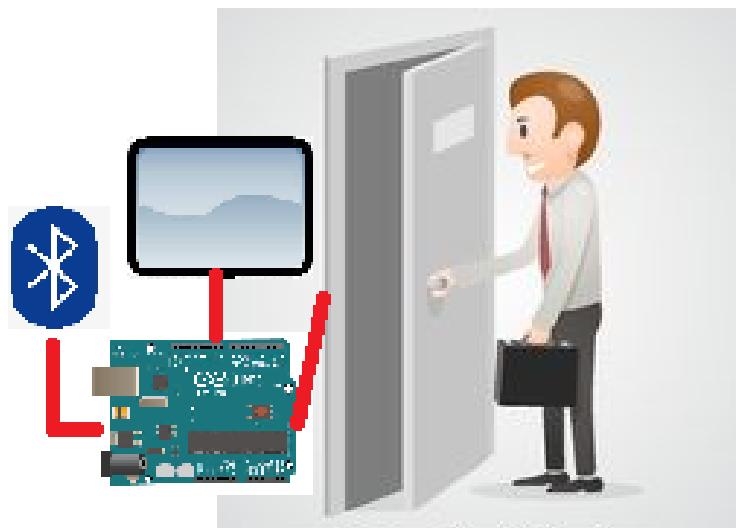
1 Indledning	3
2 Krav	4
3 Aktørbeskrivelse	4
4 Use Case beskrivelse	5
4.1 UC 1 - ADD DEVICE	5
4.2 UC 2 - REMOVE DEVICE	5
4.3 UC 3 - LOCK ON / LOCK OFF / LOCK AUTO	6
5 System beskrivelse	7
5.1 Blokbeskrivelse BA-TA	7
5.2 Internal Block Diagram for BA-TA	8
6 System Arkitektur	9
6.1 Entity Relationship Diagram	9
6.2 Sekvensdiagrammer	10
7 Brugergrænseflade	13
8 Drivers	19
8.1 Display driver	19
8.2 Touch driver	22
8.3 Bluetooth modul	22
8.3.1 Protokol	23
8.3.2 Implementering	24
8.4 SystemMaster	25
9 Tekniske overvejelse og valg	27
9.1 TFT Display	27
9.2 Brug af globale variabler	27
9.3 Bluetooth Modul	27
9.4 Touch.c og countingmillis.c	28
10 Tilegnelse af viden	29
10.1 Moduler, blokke og protokoller	29
11 Test resultater	30
12 Konklusion	31
13 Ansvarsområder	32

Figurer

1	Konceptbillede	3
2	Aktør kontekst diagram	4
3	Use Case 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	19
21	The Dot Factory karakter	20
22	Bluetooth modulet: HC-05 GW-040.	22
23	Kommunikationseksempel med AT-kommandoer ved brug af en UART forbindelse til Bluetooth modulet.	23
24	Registrering af det returnede "OK" fra Bluetooth modulet	24
25	All rå adresse information modtaget fra Bluetooth modulet	24
26	Brug af "delimeter" princippet ved brug af strtok_r funktionen	25
27	Koden til default staten	26
28	Prototype af BA-TA	31

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 gruppen 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. Hermed slipper brugerne for at skulle huske på dette, samt at skulle fumble med nøglerne når man kommer hjem med flere poser i hænderne fra dagens indkøbstur.



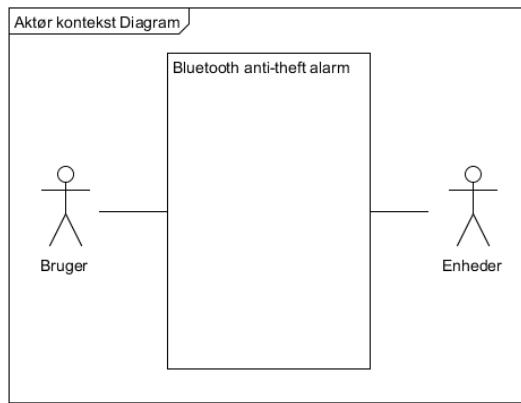
Figur 1: Konceptbillede

2 Krav

I dette afsnit beskrives kravene til hvilken funktionalitet systemet har.

- Systemet skal kunne finde de 4 sterkeste Bluetooth signaler.
- Systemet skal kunne køre uafbrudt
- Systemet skal scanne efter Bluetooth enheder hvert 5. sekund i staten ”LOCK AUTO”.

3 Aktørbeskrivelse

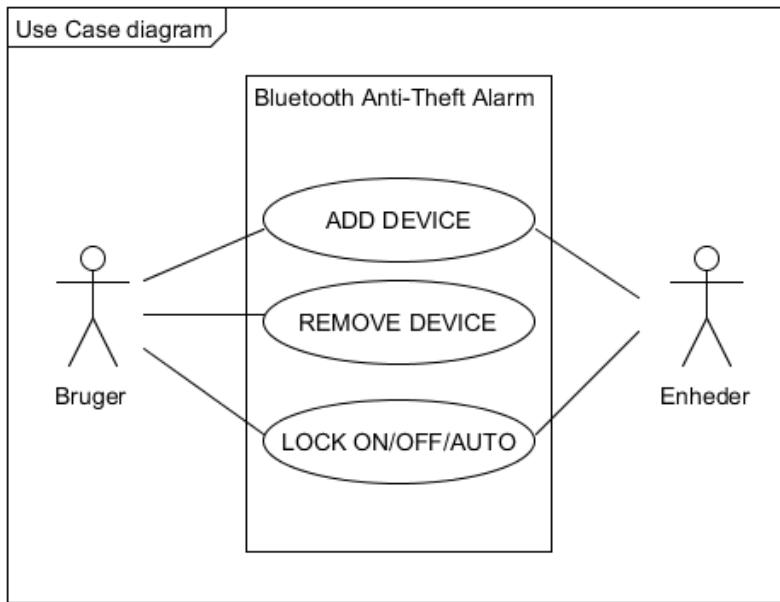


Figur 2: Aktør kontekst diagram

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: Use Case diagram

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

4.1 UC 1 - ADD DEVICE

- Brugeren trykker på ”ENTER” når pilen 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.

4.2 UC 2 - REMOVE DEVICE

- Brugeren trykker på ”ENTER” når pilen 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.

4.3 UC 3 - LOCK ON / LOCK OFF / LOCK 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

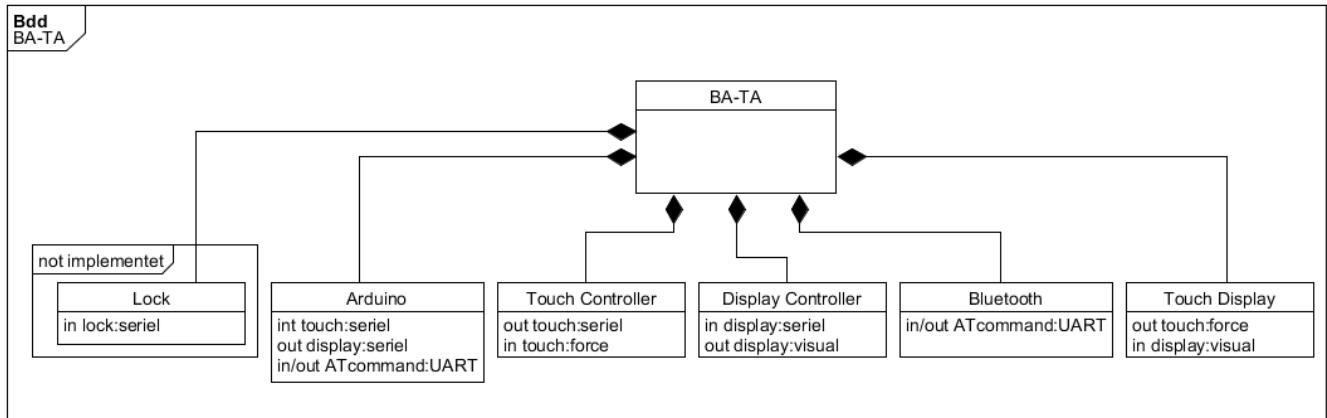
BA-TA's (Bluetooth Anti-Theft Alarm) primære funktionalitet ligger i hhv. at kunne låse op og låse hoveddøren, så snart systemet registrerer at husets ejer (og dermed smartphone) eller en anden godkendt Bluetooth-enhed, er i nærheden eller ej.

Hele systemet styres fra en samlet enhed, som brugeren kan interagere med 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 igen låst op.

I under-afsnittende herunder bliver systemets blokke og de interne forbinder præsenteret. 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. Arduino 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 initialisere og kontrollere UART driveren og Bluetooth modulet.

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

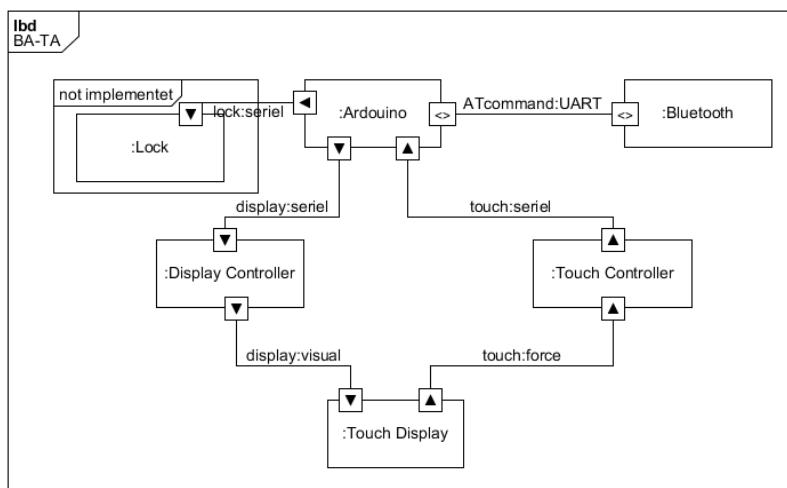
Touch Display fungerer som brugergrænseflade og er dermed måden brugeren kan interagere med systemet.

Lock

Lock er den fysiske lås, denne er ikke implementeret i projektet, men simuleres med en visuel lås på skærmen.

5.2 Internal Block Diagram for BA-TA

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



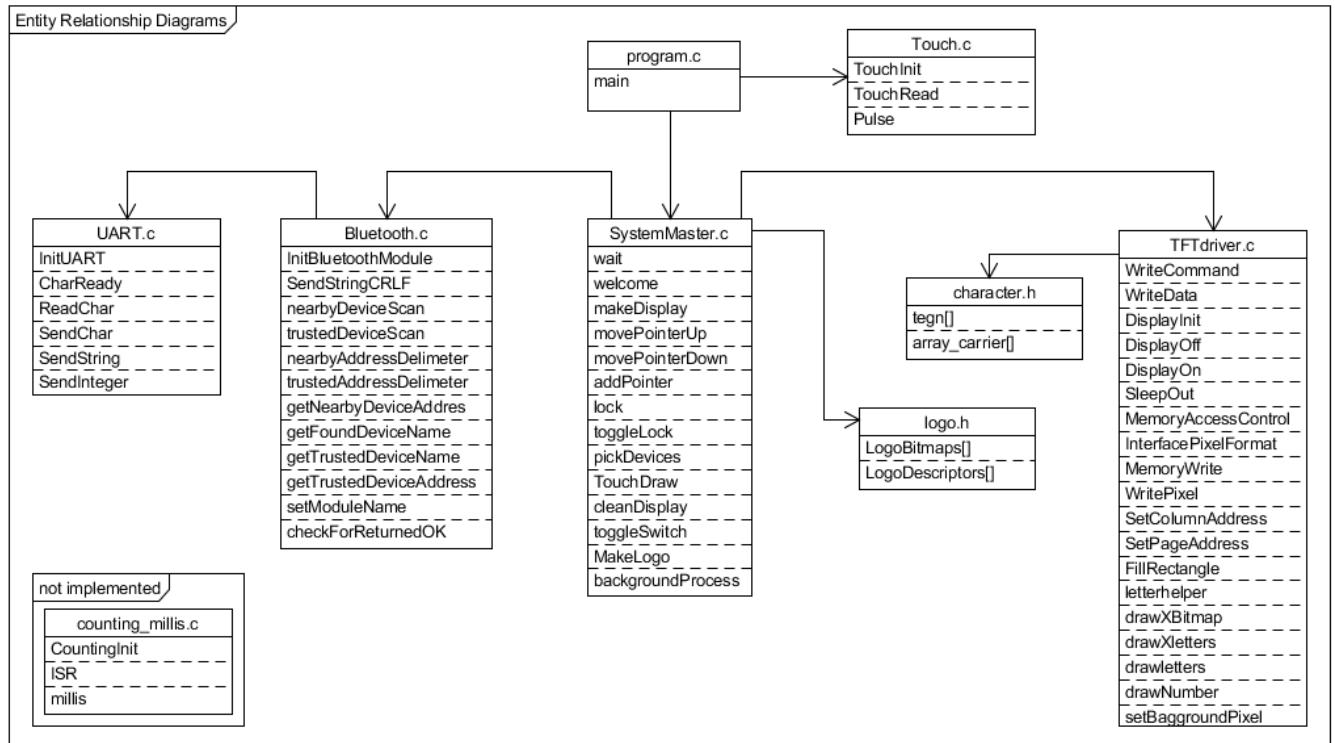
Figur 5: Ibd af BA-TA

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 rammen 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, dannes der et overblik over den logiske struktur på tværs af systemet, som er udmundet i funktioner i de forskellige driver-klasser.

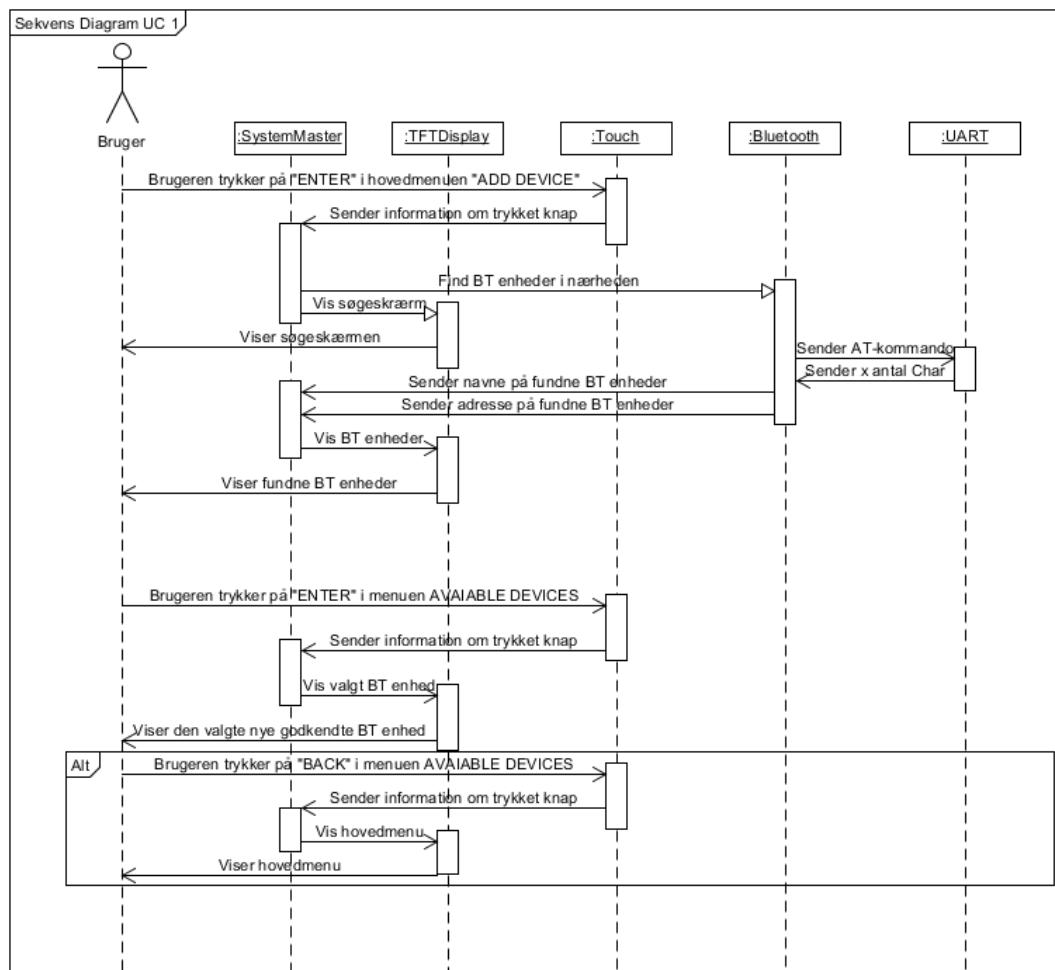


Figur 6: Entity Relationship Diagrams

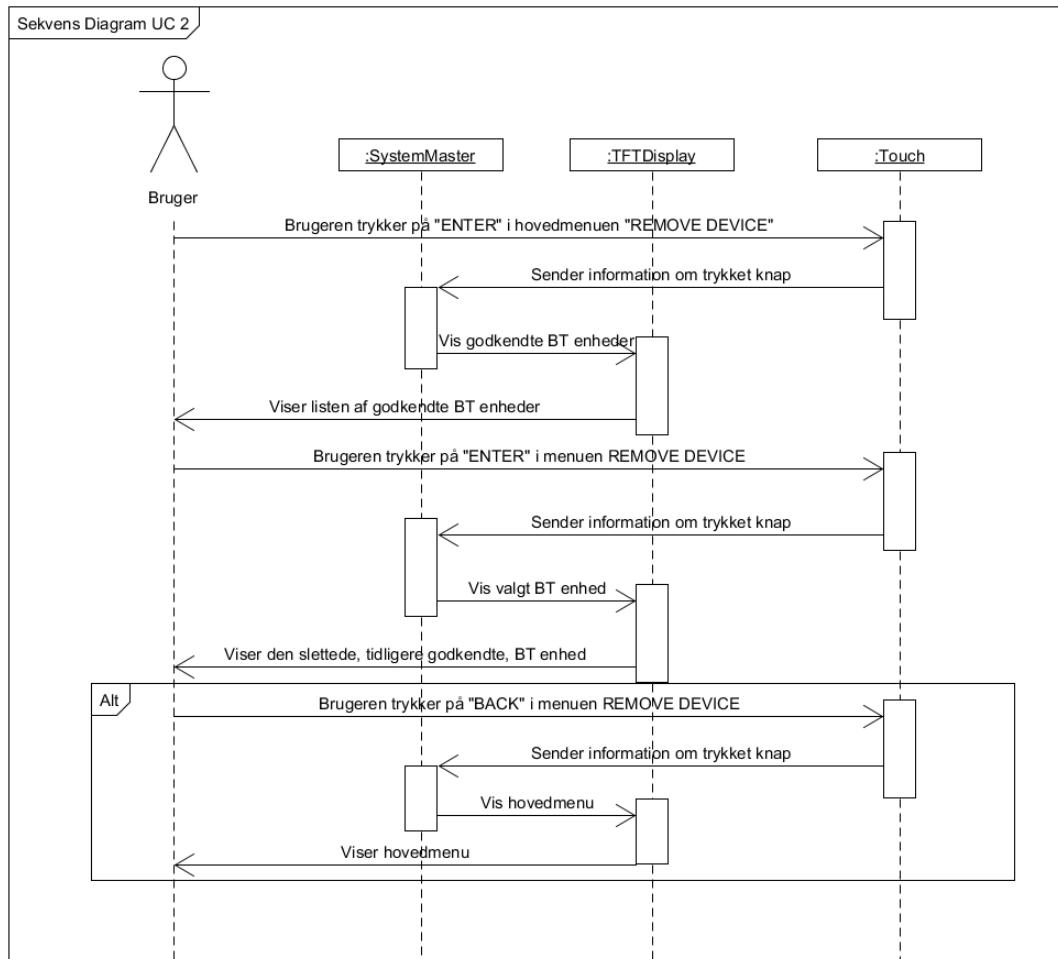
6.2 Sekvensdiagrammer

Da der hermed er dannet et overordnet overblik af systemets logiske funktionalitet, kan der dermed refereres til den logiske implementering af projektets tre Use Cases.

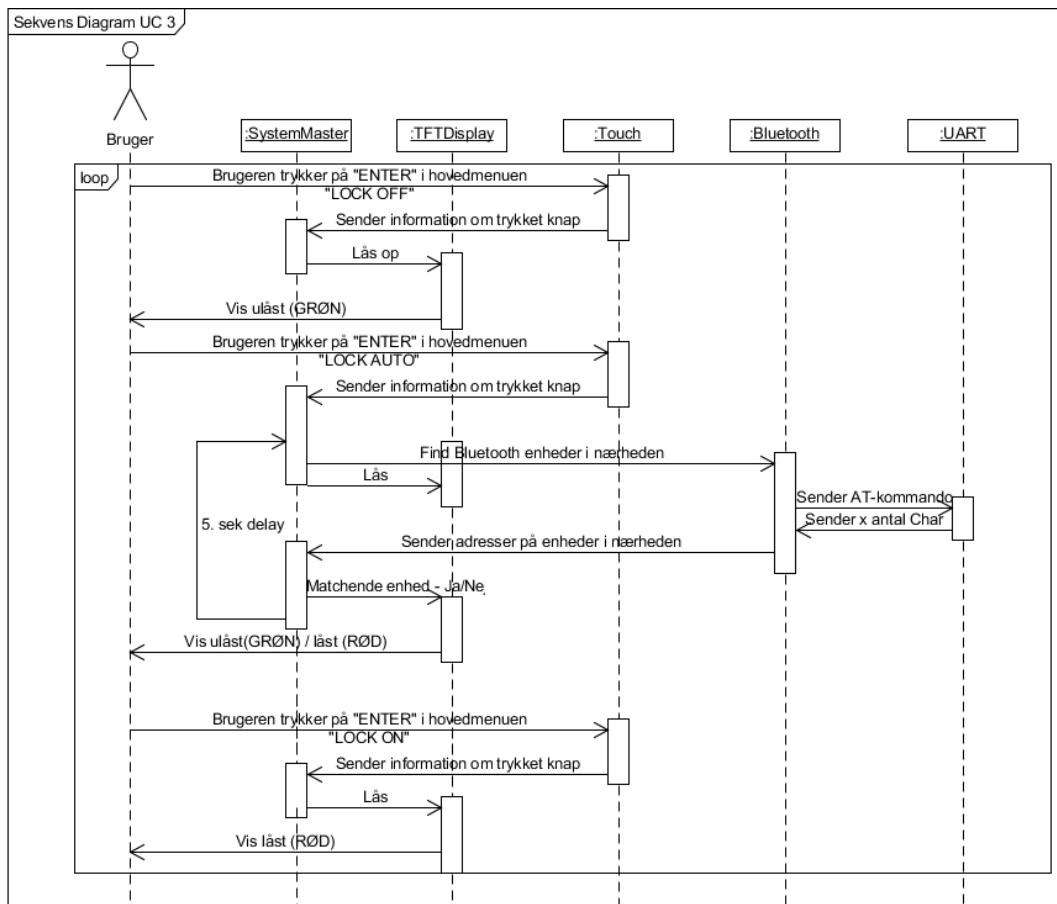
For besparelse af tekstfylde i diagrammerne og dermed også for overskuelighedens skyld, er Bluetooth nævnt i diagrammerne som forkortelsen "BT".



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 ”Welcome”. Her går systemet igang med at initialisere de forskellige moduler og drivere, 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 systemets logo.



Figur 11: Hovedmenu

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



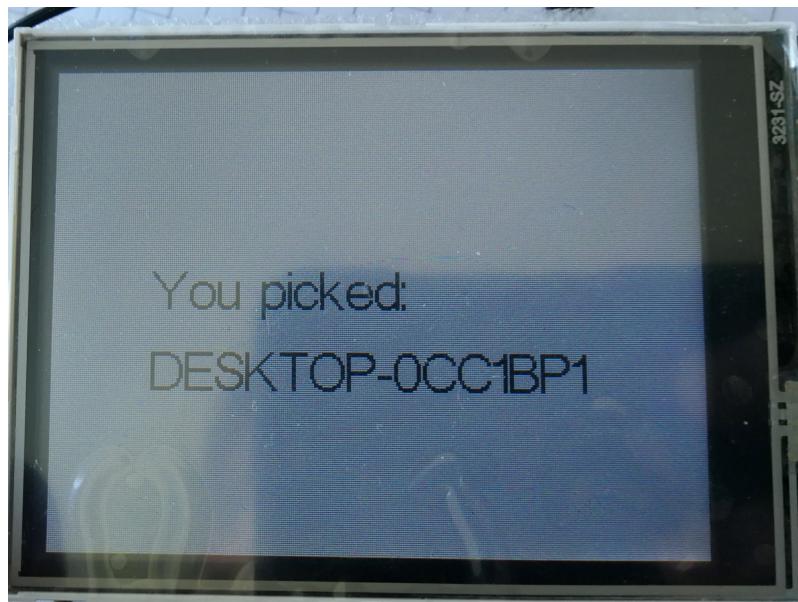
Figur 12: Searching

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.

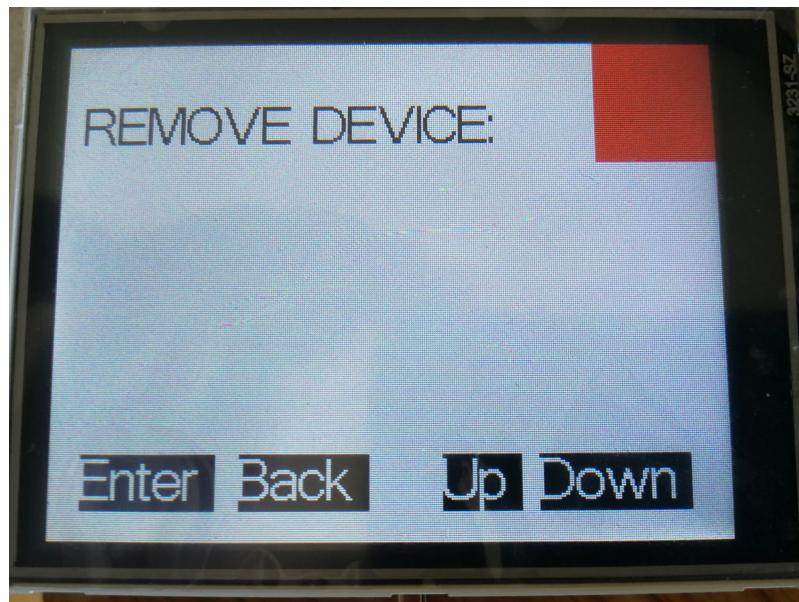


Figur 14: Valg af enhed

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 figur 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 låsens indikator 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 laver en ”UPDATE” hvert 5. sekund, og skifter låsen alt efter om der kan findes en Bluetooth-enhed, som også er på den godkendte liste. Dette ses på figur 19. Hvis der trykkes én gang til på sidste state, ændres låsens tilstand til at være manuelt låst.



Figur 19: Lås Automatisk

8 Drivers

I dette afsnit beskrives de forskellige drivere som er lavet til projektet. Driverne er delt op i flere forskellige .c filer, dette er gjort for at gøre koden mere overskuelig, og gøre funktionaliteten mere effektiv. Herunder forsøges der at gøre et overblik over de forskellige .c filer og deres interaktion.

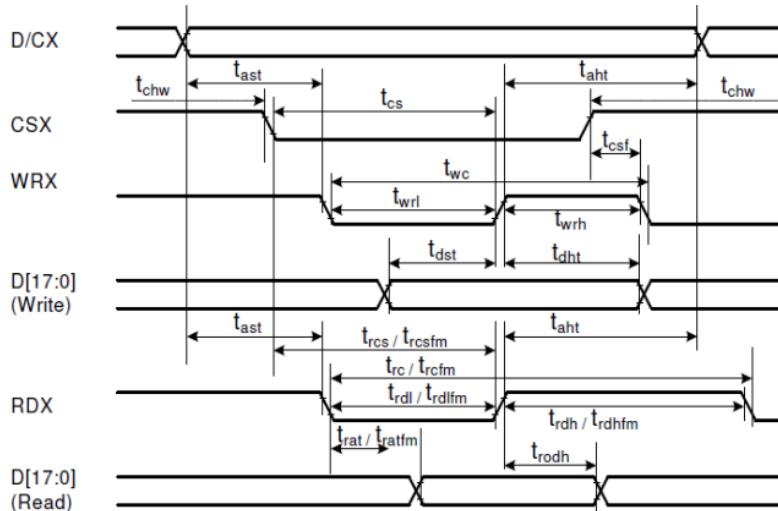
8.1 Display driver

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 selvsamme driver. .c 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, bliver der herunder 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 et reset, 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).

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 er de essentielle funktioner delt op i hver 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 [ILI9341](#) 8.2.20. og 8.2.21.:

Nogle af funktionerne i driveren, bruges ikke i dette produkt, da flere af dem er brugt til test undervejs, for at kunne debugge den givne programstuktur.

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 kan lave en firkant i en bestemt farve ud fra parameteren af funktionskaldet. Fra input vælges startx og starty samt længde og bredde. Derefter laves en firkant 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 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 forløber til til drawXletter. I denne funktion, findes længden på det givne tegn, som skal skrives. Herudover findes lægden på 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, array_carrier, er et array som indeholder information om tegn[], dette er lavet ud fra TheDotFactory[1]. 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 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 (column) 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.

8.2 Touch driver

Da systemet indebærer interaktionen fra en bruger, er der valgt at inkludere en touch driver som gør brug af **XPT2046** Touch Screen Controller, som allerede var en del af **ITDB02** Arduino MEGA shield, som også bliver brugt i Display driveren.

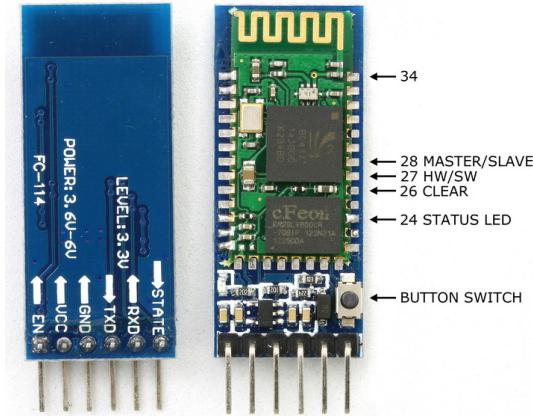
Driveren har tre funktioner, hvor Init() sætter de forskellige porte til hhv indgange og udgange, og derefter sætter de respektive ben til enten høj og lav. Funktionen pulse() står for at lave en puls på clk benet som er opsat ift. Figur 15 i **XPT2046** datasheet. Herunder ses der en tabel over den 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 seriell linje fra touch controlleren. Denne værdi skal ligges korrekt ind i en int, og bliver lagt en til hver bit, hver eneste gang der modtages et ettal på PINE bit 5, herefter returneres int værdien.

8.3 Bluetooth modul

Bluetooth-modulet der bliver brugt i BA-TA systemet er af typen HC-05 GW-040. 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 på undersiden 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.

8.3.1 Protokol

Der kommunikeres til systemets Bluetooth-modul vha. en UART-forbindelse. Gruppens implementerede UART-driver på Arduino'en gør det muligt, at Arduinoen og Bluetooth-modulet kan kommunikere sammen ved brug af AT-kommandoer. AT-kommandoerne (Attention-commands), som Bluetooth-modulet understøtter, kan bruges til generel opsætning af modulet, sætte Master/Slave roller, scanne for nærliggende Bluetooth-enheder, samt at spørge om navn på fundne Bluetooth-enheders adresser og mange andre muligheder. 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 beskeden 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?indsæt,Adresse,Her" – Bluetooth-modulet spørger om navnet på adressen til den respektive indsatte 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 nærliggende Bluetooth enheder den kan finde. Herefter svarer den tilbage med et "+INQ:Adresse;PåFundet;Enhed" 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.

8.3.2 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 instruktion/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}*8@e${h7F}iyiie{hFFFFFFFFFFFFF8E}=if ÈúiC{h7F}µùi{hFFFFFFFFFFFFF97}+~{{h1D}i{h7F}{hFFFFFFFFFFFFF9F}
Bup{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 højst 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” princippet. 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 *delimeteComma = ",";  
  

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,delimeteComma, &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, delimeteComma, &nearbySaveptr1); //Indeholder tredje del af adressen = 96BCDF
}

```

Figur 26: Brug af ”delimeter” princippet ved brug af strtok_r funktionen

På ovenstående figur, 26, kan vi se det ”delimeter” princippet 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 addressen 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. Funktionen addressDelimeter bruges derimod til når låse staten er sat til auto, da funktionaliteten i de 2 funktioner varierer i forhold til hinanden.

8.4 SystemMaster

Dette driver/modul, står for at forbinde de andre .c filer, og for at køre systemets funktionalitet. Derfor kaldes denne klasse også for master, da den fungerer som master ift. de andre klasser, som derved er slaves. Ift figur 6, ses Touch.h ikke som implementeret i denne klasse. Dette skulle den have været, men pga tidspres blev dette nedprioriteret, derudover skulle klassen counting_millis.c også være implementeret, mere om det i afsnittet Tekniske

overvejelse og valg. Herunder dannes der et overblik over koden:

Strukturen i koden er bygget op af forskellige states, som systemet kan komme i. I hver state, bruges flere forskellige funktioner, for at skabe det display som vist i afsnittet Brugergrænseflade. På figur 27 vises et udklip fra default staten, som laver Hovedmenuen fra afsnittet Brugergrænseflade.

```
default:  
    cleanDisplay();  
    pointer_max = 3;  
    drawletters("ADD DEVICE", start_position_x,start_position_y,0,0,0);  
    drawletters("REMOVE DEVICE", start_position_x,start_position_y+30,0,0,0);  
    toggleSwitch();  
    addPointer();  
    lock();  
    TouchDraw();  
    MakeLogo();  
    pointer_position_y = 0;  
    break;
```

Figur 27: Koden til default staten

Funktionaliteten fungerer på den måde, at hvis ”Lock Auto” er sat i Hovedmenuen, vil der tjekkes for gemte enheder ift nye adresser som modtages fra Bluetooth modulet. Det forløbige produkt, har en update state, som vises når den opdaterer nye værdier, og derved kan brugeren ikke integere med systemet samtidig. Dette kan ses på figur 19.

9 Tekniske overvejelse og valg

Da projektets omfang og opbygning har været meget frit, er der også blevet lavet nogle valg og fravalg i process fasen. Dette gives der et overblik over i dette afsnit, samt at forklare fordele og ulmepor ved hvert modul vi har valgt og driverne dertil.

9.1 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, at gruppen har arbejdet 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. Ulemper 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 controller shielded, ITDB02, havde en Touch controller del, blev Alphanumeric display'et valgt fra.

9.2 Brug af globale variabler

Igennem de forskellige drivere gøres der brug af globale variabler. Dette er gjort for at simplificere udviklingen af driverne. Denne programmerings-praksis er dog lidt ilde set på og er generelt ikke god programmerings-skik. Dette ville blive prioritert ved fremtidigt arbejde, således at disse globale variabler ville blive erstattet med get() og set() metoder.

9.3 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 og debugging på hvorfor den ikke kunne sættes op til at søge efter andre enheder, viste det sig at det udleverede modul ikke var det rigtige modul som gruppen oprindeligt 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 anderledes HC-05 med et større breakout-board, hvilket efter nogle få timers debugging og troubleshooting måtte konkluderes at være i stykker. Det lykkedes dog igennem Embedded Stock at få lånt et nyt HC-05 modul med et anderledes 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.

9.4 Touch.c og countingmillis.c

Da tiden var knap, blev TouchRead funktionen implementeret i main. Denne funktionalitet og derved styring af input fra brugeren, ville i et færdig produkt skulle ligge i SystemMaster. Derudover blev der taget et valg om at indsætte et delay på 100 mikrosekunder, for at sikre at systemet ikke registrerer et "dobbelttryk" fra brugeren når man bruger touch displayet. En smartere måde at sikre dette på var ved at introducere et interrupt som først triggerede når brugeren trykkede på skærmen. Det føromtalte delay, blev derudover brugt til at opdatere systemet hver 5. sekund. Dette kunne også gøres smartere, da vi ikke rammer præcis 5 sekunder, da de forskellige instruktioner som sker under delay ikke er talt med. Dette kunne være løst med klassen counting_millis, som blev lavet på baggrund af tidligere brug af funktionen millis() i Arduino's IDE. Denne funktion introducerer en realtids timer til systemet, så timingen bliver præcis 5 sekunder. Denne funktion gjorde dog brug af interrupt, som forstyrrede resten af systemet.

10 Tilegnelse af viden

Den viden som har dannet grundlag for dette projekt er tilegnet gennem AMS kurset og igennem tidligere fag på ASE, hvor embedded software har været i fokus. Herunder gives der et overblik over hvor gruppen yderligere har tilegnet sig den nødvendige viden for at kunne realisere implementeringen af systemet og den endelige prototype.

10.1 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 som udgangspunkt til at udvikle ud fra. Derimod er modulet meget kendt og er brugt af mange mennesker. Det afspejler sig i de mange forskellige websider 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 [Marty N Currey's](#) projekt har været meget brugbar at kunne følge.

TFTdisplay og Touch Disse to driver klasser er lavet ud fra undervisningen i AMS, og dertilhørende viden igennem oplæg fra underviser. Derudover er der fundet inspiration og hjælp fra [RinkyDinkyElectronics](#), hvor igennem læsning af kode er givet en lille forståelse. Derudover er alt den viden som er brugt til opbygning af disse klasser bygget på de forskellige datasheet: [ILI9341](#) og [XPT2046](#)

11 Test resultater

Prototypen til BA-TA systemet er gennemgået test, som er prædefineret af de 3 Use Cases. Herunder, i tabel 13, kan resultatet af de 3 forskellige tests ses og dermed hvorvidt resultatet af testen er godkendt eller ej.

Yderligere er der tilknyttet eventuelle bemærkninger til testens forløb og resultater.

Test resultater af BA-TA prototypen		
Use Case #	Godkendt eller ikke-godkendt?	Bemærkninger
#1	Godkendt	Test af Use Case 1 gik som forventet. En fundet BT-enhed kan tilføjes til den godkendte liste.
#2	Godkendt	Test af Use Case 2 gik som forventet. En tidligere godkendt BT-enhed kan fjernes fra den godkendte liste.
#3	Godkendt	Test af Use Case 3 gik som forventet. Alle 3 lock/låse states virker. I tilfælde af at en godkendt enhed på listen vælger at skifte navn på den respektive enhed, kan BA-TA på nuværende tidspunkt ikke opdatere til det nye navn i listen over godkendte enheder.

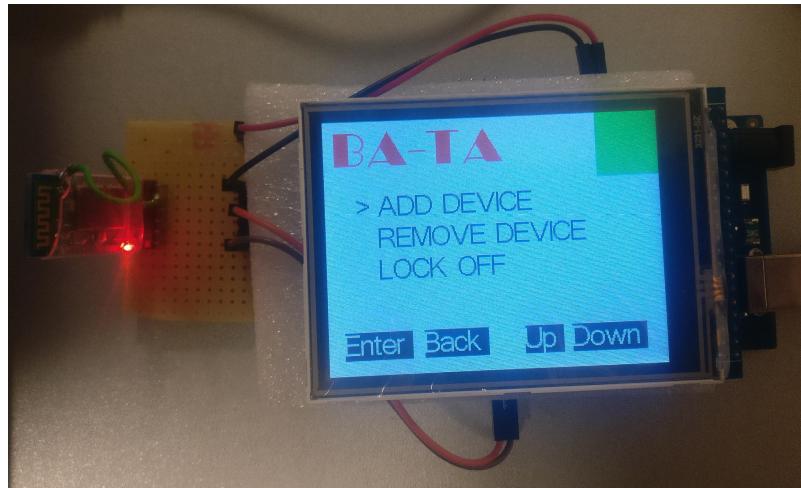
12 Konklusion

Der er i dette projekt blevet udviklet en prototype af en Bluetooth-baseret tyveri-alarm, af navnet BA-TA. Prototypen kan gøre hverdagen nemmere for husets beboere, da man slipper for at have nøglen frem hver gang der skal låses eller låses op. Systemet har en godkendt liste af enheder, som systemet i Lock/lås auto state tjekker om er hjemme eller ej. Såfremt den ikke finder nogle af de godkendte enheder i nærheden bliver dørene i huset automatisk lås, som i denne prototype er grafisk implementeret i hovedmenuen. Systemet er yderst brugervenligt og kan håndteres ved brugerinteraktion igennem brugergrænsefladen.

Undervejs i projektet er der formået at omsætte den kendte teori og viden fra undervisningen, samt at hente ny information og viden fra forskellige kilder på internettet, datablade og lignende.

En endelig udvikling og lancering af et produkt baseret på denne prototype vil dog kræve nogle modificeringer og tilføjelse af funktionalitet, som kan kategoriseres som fremtidigt arbejde.

Medlemmerne af projektgruppen er rigtig tilfredse med prototypen, dens samlede funktionalitet og test-resultater, og mener at vi har opfyldt kravene til hhv. projektet og denne rapportens omfang.



Figur 28: Prototype af BA-TA

13 Ansvarsområder

Ansvarsområder	
TFTDisplay.c	Søren
SystemMaster.c	Søren
Bluetooth.c	Philip
UART.c	Philip
Touch.c	Søren
counting_millis.c	Søren
program.c	Søren og Philip

References

- [1] TheDotFactory
TheDotFactory
- [2] Touch controller Datablad
XPT2046
- [3] Display controller Datablad
ILI9341
- [4] Rinky Dinky Elektronik
RinkyDinkyElektronik