

AARHUS UNIVERSITET

ANVENDTE MICROCONTROLLER SYSTEMER

6. SEMESTER

AMS projekt

Gruppemedlemmer:

Søren Landgrebe
Philip Nygaard Scmhidt

Studienr:

201508295
201506381



27. maj 2018

Indhold

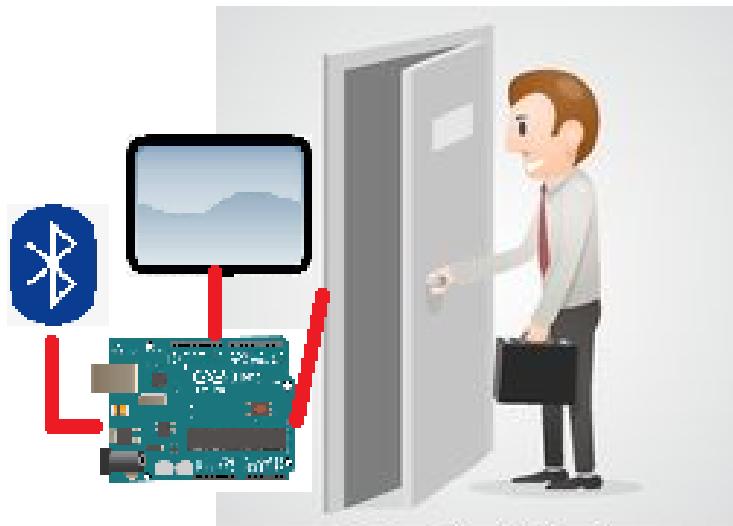
1	Indledning	3
2	Krav	4
3	System beskrivelse	5
3.1	Blokbeskrivelse BA-TA	5
3.2	Internal Block Diagram for BA-TA	6
3.3	Entity Relationship Diagram for BA-TA	6
4	Bruger Grænse Glade	7
5	Graphic display driver	13
6	Display driver	14
7	Touch driver	17
8	Fordele og ulemper	18

Figurer

1	Konceptbillede	3
2	Bdd af BA-TA	5
3	Ibd af BA-TA	6
4	Entity Relationship Diagrams	6
5	welcome	7
6	Hoved Menu	8
7	Searching	8
8	Tilgængelige enheder	9
9	Valg af enhed	9
10	Fjern en enhed	10
11	Ingen enheder	10
12	Valgt fjernet en enhed	11
13	Låst op	11
14	Lås Automatisk	12
15	Bus Timing	14
16	The Dot Factory karakter	15

1 Indledning

Næsten alle danskere har nu til dags en mobil på sig som, som indeholder et bluetooth modul. Dette vil vi gerne udnytte til at kunne låse og åbne hoveddøren hos private personer. Dette betyder at forbrugeren aldrig skal tænke på at have nøgle med fra hjemmet, eller finde dem inden man kan lukke sig selv ind. Denne prototype BA-TA (Bluetooth anti-theft alarm) vil derfor kunne låse og låse op for en dør, alt efter om man med sin bluetooth enhed er i nærheden af døren. Hele systemet styres fra en samlet enhed, som kan integreres fra brugeren gennem en skærm og dertilhørende touch funktion.



Figur 1: Konceptbillede

Igennem brugergrænsefladen kan brugeren tilføje og fjerne enheder (personer) som skal kunne låse og låse op for døren. Når den sidste enhed er uden for rækkevidde, låses døren, og så snart systemet ser en godkendt enhed indenfor rækkevidden bliver døren låst op.

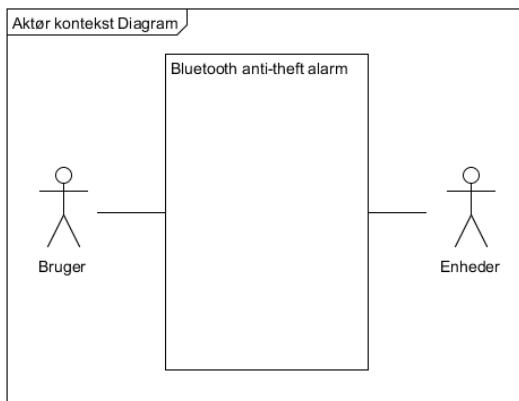
2 Krav

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 sterkeste bluetooth signaler.
- Systemet skal kunne køre ”hele tiden”
- Systemet skal opdateres en gang i hvert sekund.

3 Aktørbeskrivelse



Figur 2: Aktør Kontekst diagram

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

Primær

Bruger: Den aktør der interagerer med systemet og vælger den ønskede funktionalitet

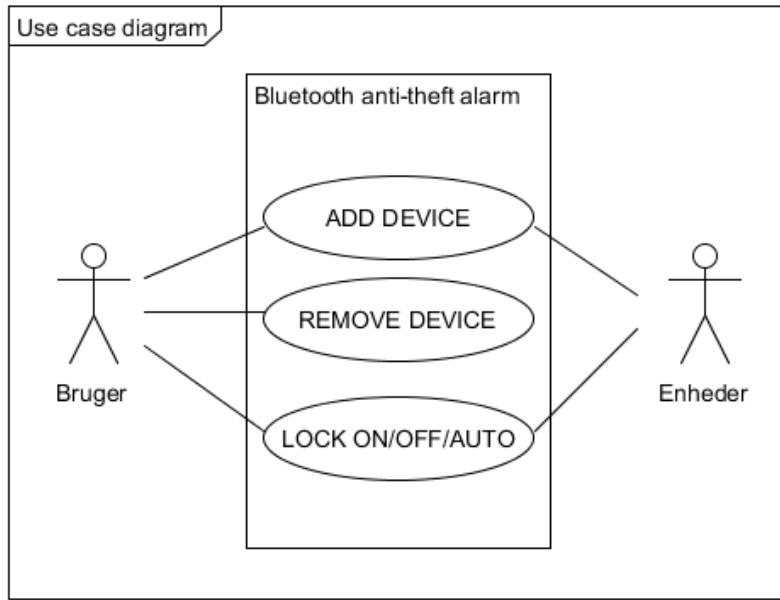
Recording system: Den aktør der modtager det endelige produkt

Sekundær

Mic støj: Et input til det samlede system

Mic tale+støj: Et input til det samlede system

4 Use case beskrivelse



Figur 3: Usecase diagram

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

4.0.1 UC 1 - Turn on filter

Brugeren trykker på SW1 og filteret aktiveres.

4.0.2 UC 2 - Turn off filter

Brugeren trykker på SW1 og filteret deaktiveres.

4.0.3 UC 3 - Filter aktiv

Recording system modtager detfiltrede lyd, hvis prækonditionen UC 1 er udført.

5 Ikke-funktionelle krav

Kravene er delt op i tre underkategorier. Krav der relaterer til problemet. krav der relaterer til DSP platform og algoritme. Til sidst er der en kategori der beskriver kravene til systemet på baggrund af de to første kategorier.

5.1 Problemrelateret krav

1. R1: Systemet skal have 2 mikrofoner og 1 højtaler
2. R2: Filteret skal gøre brug af LMS algoritmen
3. R3: Systemet skal kunne processerer lyd i frekvensbåndet 50-20000Hz.
4. R4: Systemet skal kunne dæmpe uønkset støj 30dB.
5. R5: Systemet skal kunne dæmpe støj uden at dæmpe ønsket lydsignal.
6. R6: Systemet burde have en latency under 30ms.
7. R7: Systemet burde have et dynamikområde på min 80dB

5.2 System og algoritme krav

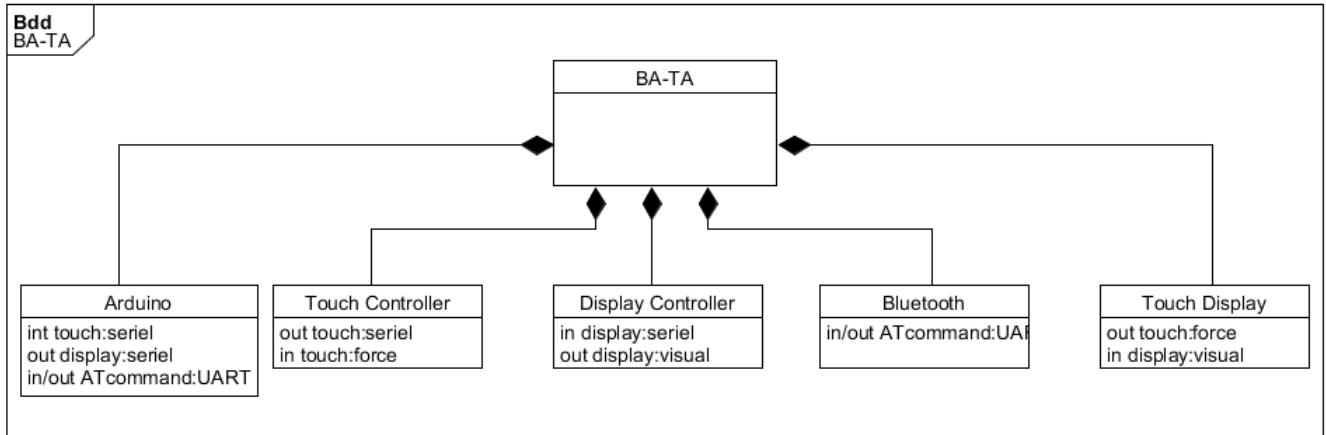
8. R8: Filter algoritmen skal implementeres med fixed point
9. R9: Filteret skal max bruge 10kByte memory
10. R10: Filteret skal implementeres på Blackfin BF533
11. R11: Filteret må max benytte 98% DSP load

5.3 Afledte krav

12. DR1: DSP systemet skal kunne håndtere en samplingsrate på min 44100kHz(På baggrund af krav R3)
13. DR2: Filteret skal implementeres med 1.15 fixed point.(På baggrund af krav R7 og R8. Dette giver et dynamikområde på 96dB)
14. DR3: Filter latency må max forsinkes 1280 samples(På baggrund af R6. $(1/44100)*1280=30ms$)
15. DR4: Filteret må max bruge 13333 cycles af DSP processering for hver sample. (På baggrund af R10, R11 og DR1 $(600MHz/44.1kHz)*98\%$)

6 System beskrivelse

I dette afsnit laves et overblik over systemets blokke og de interne forbinder. Derudover gives et overblik over funktionerne som er implementeret i projektet. På figur 2 ses et overordnet Bdd for projektet, hvor de interne forbinder forklares på figur 3.



Figur 4: Bdd af BA-TA

6.1 Blokbeskrivelse BA-TA

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

Arduino

Arduino står for at styre hele systemet. Den skal styre alt funktionalitet som projektet skal udføre, og står som master ift. resten af projektet. Arduino blokken står for at initialisere controller blokkene, hhv. Touch og Display, derudover står den for at modtage signaler fra touch, og sende signaler til display. Derudover står den for at få bluetooth til at sende de respektive værdier.

Touch Controller

Touch Controller står for at modtage værdier fra Touch Display, sende værdien videre til Arduino

Display Controller

Display Controller står for modtage en værdi fra Arduino og sende sende værdien til Touch Display.

Bluetooth

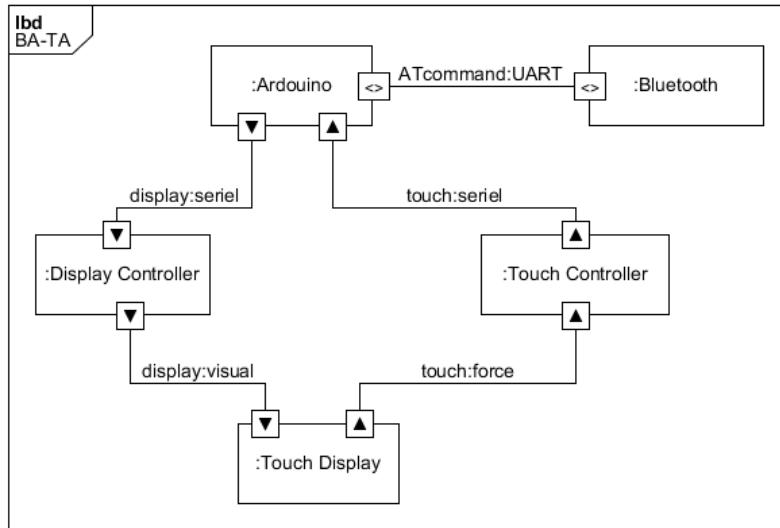
Er et bluetooth modul, som vha. AT kommandoer kan kommunikere med Arduino.

Touch Display

Touch Display fungerer som brugergrænseflade og integering mellem system og bruger.

6.2 Internal Block Diagram for BA-TA

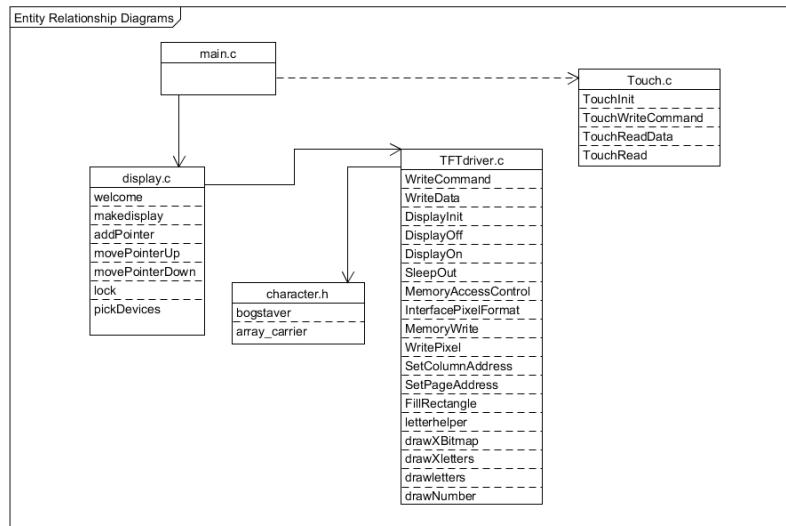
På figur 3 ses et overordnet IBD over selve systemet, som er bygget på Bdd'et.



Figur 5: Ibd af BA-TA

6.3 Entity Relationship Diagram for BA-TA

Herefter gives et overblik på figur 4 over koden til driverne som er skrevet til dette projekt.



Figur 6: Entity Relationship Diagrams

7 Bruger Grænse Flade

I dette afsnit gives et overblik over brugergrænsefladen og integeringen fra brugeren. Afsnittet er bygget op af billeder, hvor hvert billede viser brugergrænsefladens visuelle struktur.

Første display man møder er velkommen.



Figur 7: welcome

Næste display er en hovedmenu, hvor der er tre forskellige muligheder, som ses på figur 6. Hertil tilføjes 4 trykknapper, hvor brugeren kan integere med systemet. Med "Enter" vælger brugeren en funktion. "Back" gør brugeren i stand til at gå tilbage til hovedmenuen, hvis man åbner en af de kommene states. "Up" og "Down" flytter pointeren hhv. op og ned. Herudover laves en lås øverst i højre hjørne. Denne lås viser grøn ved låst op og rød ved låst. Øverst til venstre på skærmen vises projektets logo.



Figur 8: Hoved Menu

Hvis brugeren trykker på ”ADD DEVICE” søger BA-TA efter de 4 stærkeste Bluetooth signaler i nærheden.



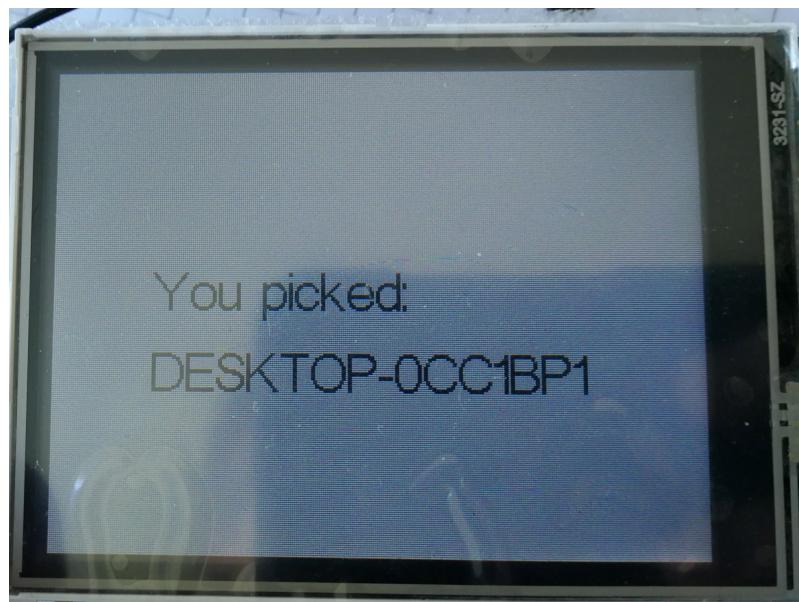
Figur 9: Searching

Herefter vises de enheder som kan vælges til systemet. Dette ses på figur 8.



Figur 10: Tilgængelige enheder

Hvis der vælges en enhed, viser displayet hvilken enhed der er valgt.

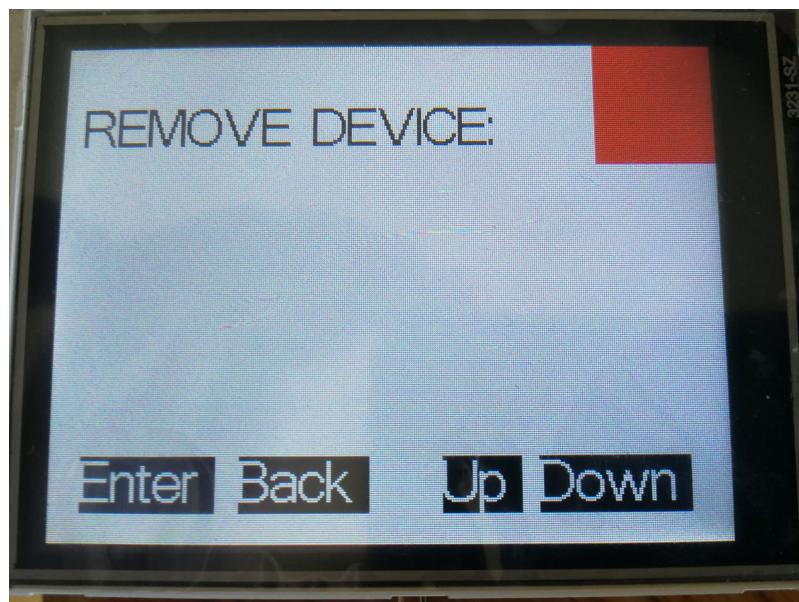


Figur 11: Valg af enhed

Hvis brugeren herefter vælger funktionen ”REMOVE DEVICE” fra figur 6, åbnes hvilke enheder der er gemt Dette ses på 10.. Hvis der ikke er gemt nogen device, er displayet tomt, som vises på figur 11.



Figur 12: Fjern en enhed



Figur 13: Ingen eneheder

Heresfter kan brugeren vælge at slette en bestemt enhed. Dette sker ved at trykke ”Enter” ved den rette enhed. Displayet viser på figur 12 en besked om hvilken enhed der er slettet.



Figur 14: Valgt fjernet en enhed

Hvis brugeren trykker på den sidste state i Hoved Menuen, låses systemet op, teksten skifter og låsen skifter farve. Dette ses på 13.



Figur 15: Låst op

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



Figur 16: Lås Automatisk

8 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

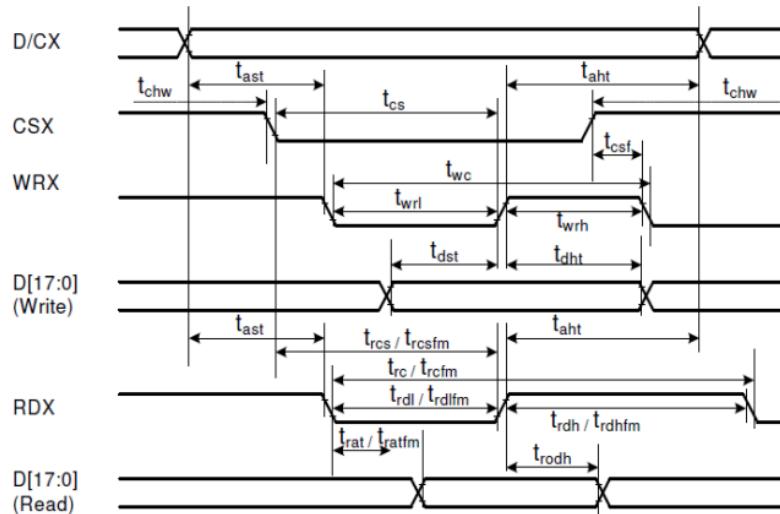
9 Display driver

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. 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 15. 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 i 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 17: 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 15. 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 karakterer: 95



Figur 18: 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)

letterhelper(numberletter, startx, starty, Red, Green, Blue)
Funktion som bruger input fra Include filen character, til at bestemme hver karakter bredde og længde
Parametre: numberletter: Bestemmer hvilken karakter der skal hentes Startx: Startværdi på x-aksen Starty: Startværdi på y-aksen Red: farve(værdi) Green: farve(værdi) Blue: farve(værdi)

drawXBitmap(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

drawletters(str[],startx, starty,Red, Green, Blue)

Funktion som modtager en en streng, og konverterer ascii værdien til den rette værdi iif 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)

drawNumber(number,startx, starty,Red, Green, Blue)

Funktion som modtager en en integer, og konverterer ascii værdien til den rette værdi iif character.h

Paramentre:

number: Modtager et interger tal

Startx: Startværdi på x-aksen

Starty: Startværdi på y-aksen

Red: farve(værdi)

Green: farve(værdi)

Blue: farve(værdi)

setBaggroundPixel(int red, int green, int blue)

Funktion som sætter baggrundsfarven af tekst og tal

Paramentre:

Red: farve(værdi)

Green: farve(værdi)

Blue: farve(værdi)

10 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.

11 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 ulmepor 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 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. 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 brug af globale variabler, dette er gjort da for at simplificere udviklingen af driverene. Denne metode at programmere på er dog ikke god program skik, og hvis gruppen havde afsat mere tid til at udvikling, ville det første være at erstatte disse globale variabler, med get og set metoder.

Bluetooth

Litteratur

- [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.